Automatisches Modellieren von Agenten-Verhalten

Erkennen, Verstehen und Vorhersagen von Verhalten in komplexen Multi-Agenten-Systemen

Dissertation

zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften (doctor rerum naturalium)
im Fach Informatik

eingereicht an der

Mathematisch-Naturwissenschaftlichen Fakultät II
der Humboldt-Universität zu Berlin

von

Diplom-Informatiker Jan Wendler,
geboren am 21. Dezember 1973 in Berlin

Präsident der Humboldt-Universität zu Berlin
Prof. Dr. Jürgen Mlynek

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät II
Prof. Dr. Elmar Kulke

Gutachter / Gutachterinnen:

1. Prof. Dr. Hans-Dieter Burkhard
2. Prof. Dr. Hans-Joachim Lenz
3. Prof. Dr. Martin Riedmiller

eingereicht: 6. Februar 2003
Tag der mündlichen Prüfung: 26. August 2003

Anmerkungen für den Leser

In der Arbeit wird die alte deutsche Rechtschreibung (Grundfassung aus dem Jahre 1902) verwendet. Abweichend davon wird der Begriff Verhalten sowohl für die Singular- als auch für die Plural-Form verwendet. Desweiteren werden einige zusammengesetzte Worte nicht zusammen sondern mit einem Bindestrich geschrieben.

Zusammenfassung

In Multi-Agenten-Systemen (MAS) kooperieren und konkurrieren Agenten, um ihre jeweiligen Ziele zu erreichen. Für optimierte Agenten-Interaktionen sind Kenntnisse über die aktuellen und zukünftigen Handlungen anderer Agenten, der Interaktionsparter (IP), hilfreich. Bei der Ermittlung und Nutzung solcher Kenntnisse kommt dem automatischen Erkennen und Verstehen, sowie der Vorhersage von Verhalten der IP auf Basis von Beobachtungen, besondere Bedeutung zu.

Diese Arbeit beschäftigt sich mit der automatischen Bestimmung und Vorhersage von Verhalten der IP durch einen Modellierenden Agenten (MA). Der MA generiert fallbasierte, adaptive Verhaltens-Modelle seiner IP und verwendet diese zur Vorhersage ihrer Verhalten. Als Anwendungsszenario wird mit dem virtuellen Fußballspiel des *RoboCup* ein komplexes MAS betrachtet.

Nach der Zielstellung und einer ausführlichen Motivation der Verhaltens-Modellierung wird der verwendete Modellierungsansatz vorgestellt. Hierbei wird eine Strukturierung des Prozesses der Generierung (umfasst die Verhaltens-Bestimmung) und der Anwendung von Verhaltens-Modellen (hauptsächlich zur Verhaltens-Vorhersage) vorgenommen.

Die Verhaltens-Bestimmung besteht aus der Spezifikation von Verhalten durch einen Designer und der automatischen Erkennung von Verhalten durch den MA. Die Verhalten sind dabei aus Ereignissen zusammengesetzt, d.h. die Erkennung erfolgt in zwei Schritten: zuerst werden Ereignisse erkannt, aus denen dann Verhalten abgeleitet werden.

Es werden fallbasierte, teamspezifische Verhaltens-Modelle generiert und verwendet. Die Fälle eines Verhaltens-Modelles bestehen aus Paaren von Auslösern und Verhaltens-Essenzen. Ein Auslöser stellt dabei den relevanten Teil der Situation zum Start des Verhaltens dar und Verhaltens-Essenzen bestehen aus den charakteristischen Werten eines Verhaltens. Zum Vergleich zweier Auslöser bzw. Verhalten wird ein Ähnlichkeitsmaß definiert, welches teilweise automatisch an das jeweilig modellierte Team angepasst wird.

Die Verhaltens-Vorhersage wird auf Grundlage einer Situation in mehreren Schritten durchgeführt. Zuerst werden alle möglichen Auslöser dieser Situation bestimmt, für die dann die ähnlichsten Fälle in der Fallbasis ermittelt werden. Aus diesen Fällen wird ein geeigneter Fall ausgewählt, dessen Verhalten an die aktuelle Situation angepaßt wird.

Der vorgestellte Ansatz zur Verhaltens-Modellierung wird anhand einer objektorientierten Implementierung evaluiert. Die Verhaltens-Erkennung kann mit Korrektheits- und Vollständigkeitswerten von durchschnittlich über 0,98 als sehr erfolgreich eingeschätzt werden. Die Verhaltens-Vorhersage kann dagegen nur mit Einschränkungen als erfolgreich angesehen werden. Die Vorhersage-Genauigkeit konnte im Vergleich zu einer zufälligen Verhaltens-Auswahl durchgängig gesteigert werden (im Durchschnitt um den Faktor 2,5). Mit durchschnittlichen Vorhersage-Genauigkeiten von unter 0,5 bedarf der vorgestellte Ansatz aber weiterer Optimierungen.

Der Hauptbeitrag dieser Arbeit besteht in der Ausarbeitung, Realisierung und Evaluierung eines Ansatzes zur automatischen Verhaltens-Modellierung für ein komplexes Multi-Agenten-System.

Inhaltsverzeichnis

1 Einführung **1**

2 Motivation und Einordnung **5**

2.1 Zielstellung 5

2.2 Kooperation und Konkurrenz zwischen Agenten 7

2.2.1 Agenten, Interaktionspartner und Beobachter 7

2.2.2 Kooperation, Konkurrenz und Kommunikation 9

2.2.3 Kooperation ohne Kommunikation 10

2.3 Verhaltens-Modellierung 11

2.3.1 Erkennen und Interpretieren von Verhalten 12

2.3.2 Merkmale von Verhaltens-Modellen 14

2.3.3 Anwendungsmöglichkeiten von Verhaltens-Modellen . . . 18

2.3.4 Automatische Generierung von Verhaltens-Modellen . . . 22

2.4 Einordnung dieser Arbeit 23

2.4.1 Verhaltens-Bestimmung 23

2.4.2 Generierung von Verhaltens-Modellen 24

2.4.3 Anwendung von Verhaltens-Modellen 24

3 Der Modellierungsansatz **25**

3.1 Die Grundlagen für die Verhaltens-Modellierung 26

3.1.1 Diskrete Zeit 26

3.1.2 Welt-Situation, Beobachtung und Situation 26

3.1.3 Struktur der Interaktionspartner 29

3.2 Die verwendete Multi-Agenten Umgebung 30

3.3 Begriffe für die Verhaltens-Modellierung 34

3.3.1 Verhalten und Verhaltens-Muster 34

3.3.2 Auslöser und Auslöser-Muster 36

3.4 Konkretisierung der Definition von Verhaltens-Modellen 36

3.5 Generierung eines Verhaltens-Modells 37

3.5.1 Erzeugung eines Falles 37

3.5.2 Spezifikation des Ähnlichkeitsmaßes 39

3.6 Anwendung eines Verhaltens-Modells 39

3.7 Annahmen des Ansatzes 40

4 Die Verhaltens-Bestimmung 43
4.1 Verwendete Bezeichnungen . . . 43
4.2 Das Verhaltens-Muster Paß . . . 44
4.2.1 Transfer eines Balles als Basis für einen Paß . . . 44
4.2.2 Was bedeutet Ballkontrolle? . . . 46
4.2.3 Vom Balltransfer zum Paß . . . 48
4.2.4 Verschiedene Ausprägungen von Pässen . . . 51
4.3 Formalisierung der Verhaltens-Bestimmung . . . 54
4.3.1 Ereignisse als Grundbestandteil von Verhalten . . . 55
4.3.2 Genauigkeits-Stufen beim Beschreiben von Verhalten . . . 56
4.3.3 Spezifikation von Ereignissen und Verhalten . . . 57
4.3.4 Erkennung von Verhalten . . . 63
4.4 Weitere Verhaltens-Muster . . . 65
4.4.1 Das Verhaltens-Muster Dribbeln . . . 67
4.4.2 Das Verhaltens-Muster Torschuß . . . 69
4.4.3 Das Verhaltens-Muster Klären . . . 70
4.4.4 Gleiche Ausprägungen verschiedener Verhaltens-Muster . . . 72
4.4.5 Zusammenfassung der bisherigen Verhaltens-Muster . . . 74
4.4.6 Das Verhaltens-Muster Doppelpaß . . . 78
4.4.7 Das Verhaltens-Muster Abseitsfalle . . . 79
4.5 Objektorientierte Realisierung . . . 82
4.5.1 Grundlegende Anforderungen und Entscheidungen . . . 82
4.5.2 Verschiedene Zustände von Verhalten . . . 84
4.5.3 Die Klassen für Ereignis-Muster . . . 87
4.5.4 Strukturierung der Ereignisse bzgl. ihrer Nutzung . . . 90
4.5.5 Die Klassen für Verhaltens-Muster . . . 94
4.5.6 Erkennung von Verhaltens-Objekten . . . 96

5 Fallbasierte Generierung von Verhaltens-Modellen 101
5.1 Die Auslöser für die Verhaltens-Muster . . . 102
5.1.1 Vorbemerkungen zur Wahl von Auslöser-Mustern . . . 102
5.1.2 Das Auslöser-Muster für das Paß-Verhalten . . . 103
5.1.3 Das Auslöser-Muster für das Torschuß-Verhalten . . . 105
5.1.4 Das Auslöser-Muster für das Dribbel-Verhalten . . . 106
5.1.5 Das Auslöser-Muster für das Klären-Verhalten . . . 107
5.2 Formalisierung der Auslöser-Bestimmung . . . 108
5.2.1 Spezifikation von Auslösern . . . 108
5.2.2 Erkennung von Auslösern . . . 110
5.3 Die Essenzen der Verhaltens-Muster . . . 111
5.3.1 Die Essenz des Verhaltens-Musters Paß . . . 112
5.3.2 Die Essenz des Verhaltens-Musters Torschuß . . . 112
5.3.3 Die Essenz des Verhaltens-Musters Dribbeln . . . 113
5.3.4 Formalisierung der Spezifikation und der Identifikation von Verhaltens-Essenzen . . . 113
5.4 Fallbasis mit Auslöser-Verhaltens-Fällen . . . 114
5.4.1 Generierung eines Falles . . . 114
5.4.2 Integration eines Falles in die Fallbasis . . . 116

5.5 Ähnlichkeitsmaß für die Fälle . . . 116
5.5.1 Allgemeine Betrachtungen . . . 117
5.5.2 Zusammengesetztes Ähnlichkeitsmaß . . . 118
5.5.3 Lokale Ähnlichkeiten . . . 119
5.5.4 Automatische Bestimmung von Gewichten . . . 120
5.6 Objektorientierte Realisierung . . . 125
5.6.1 Attribute von Auslöser und Verhalten . . . 126
5.6.2 Auslöser und Verhalten . . . 127
5.6.3 Fälle . . . 128
5.6.4 Teammodel . . . 129

6 Anwendung von Verhaltens-Modellen 131
6.1 Vorhersage von Verhalten . . . 131
6.1.1 Erkennung von Entscheidungs-Situationen . . . 132
6.1.2 Identifizierung von Auslösern . . . 132
6.1.3 Ermittlung von Fällen mit ähnlichen Auslösern . . . 132
6.1.4 Auswahl eines Auslöser-Verhaltens-Falles . . . 133
6.1.5 Anpassung des Verhaltens . . . 134
6.1.6 Zusammenfassung der Verhaltens-Vorhersage . . . 135
6.2 Aktualisierung von Verhaltens-Modellen . . . 137
6.2.1 Aktualisierung der Fallbasis . . . 137
6.2.2 Aktualisierungen von Parametern . . . 138
6.3 Objektorientierte Realisierung . . . 138
6.4 Resümee: Fallbasiertes Verhaltens-Modell . . . 141

7 Evaluierung des Ansatzes 143
7.1 Allgemeine Bemerkungen . . . 143
7.1.1 Implementation . . . 143
7.1.2 Primärdaten . . . 145
7.2 Evaluierung der Verhaltens-Bestimmung . . . 145
7.2.1 Laufzeit der Verhaltens-Erkennung . . . 146
7.2.2 Automatische Analysen zur Abdeckung . . . 146
7.2.3 Manuelle Analysen zur Korrektheit und Vollständigkeit . 149
7.3 Evaluierung der Verhaltens-Vorhersage . . . 154
7.3.1 Genauigkeit und Wirksamkeit . . . 155
7.3.2 Bestimmung der Gewichte für Auslöser . . . 157
7.3.3 Abhängigkeit von der Anzahl der Fälle . . . 160
7.3.4 Vorhersage des Paß-Verhaltens und des Paß-Empfängers . 162
7.3.5 Weitere Analysen . . . 167
7.4 Zusammenfassung . . . 168

8 Zusammenfassung und Ausblick 171
8.1 Zusammenfassung . . . 171
8.2 Ausblick: Verhaltens-Bestimmung . . . 174
8.3 Ausblick: Generierung und Anwendung von Verhaltens-Modellen 175

A Notation verwendeter Diagramme 179

Abbildungsverzeichnis **181**

Tabellenverzeichnis **183**

Verzeichnis von Verhaltens-Mustern **184**

Literaturverzeichnis **185**

Index **191**

Kapitel 1

Einführung

Autonome Computersysteme werden in Zukunft eine zunehmende Rolle bei der Erledigung von Aufgaben in der Umwelt des Menschen spielen. Dies wird durch Fortschritte in der Hardwareentwicklung und vor allem durch neue Entwicklungen in der Künstlichen Intelligenz möglich. Beispielsweise kann sich ein Auto mit Hilfe der Systeme ALVINN [47] und MANIAC [24] selbständig über 95% des Straßennetzes der Vereinigten Staaten steuern.

Für einen Einsatz solcher Systeme im öffentlichen Straßenverkehr, reicht es nicht aus, daß sie für sich allein sicher fahren können, sie müssen auch mit anderen Fahrzeugen interagieren. Daher ist es von immer größerer Wichtigkeit, die Interaktionen autonomer „Agenten“ miteinander und deren Auswirkungen in der realen Welt zu untersuchen.

Unter einem Agenten wird ein autonomes Computersystem verstanden, welches sich in einer Umgebung befindet und zum Erreichen seiner Ziele flexibel mit seiner Umwelt interagiert. In Multiagentensystemen befinden sich mehrere Agenten in einer Umgebung, die nicht nur jeweils direkt mit ihrer Umwelt, sondern über diese auch miteinander interagieren. Dabei können Agenten gemeinsame Ziele haben und diese durch Zusammenarbeit effizienter bzw. überhaupt erst realisierbar machen (kooperative Agenten) oder entgegengesetzte Ziele verfolgen (konkurrierende Agenten).

Für optimierte Agenten-Interaktionen müssen die Agenten die Verhalten ihrer jeweiligen Interaktionspartner berücksichtigen. Während kooperative Agenten ihr Verhalten koordinieren können, indem sie Kommunikation oder vorher definierte Interaktionsprotokolle nutzen, ist bei konkurrierenden Agenten im allgemeinen keine dieser Möglichkeiten anwendbar. Für eine Berücksichtigung der Verhalten von Interaktionspartnern sind Kenntnisse über deren Verhalten notwendig. Diese Kenntnisse sollen automatisch auf der Grundlage von Beobachtungen der Handlungen von Interaktionspartnern ermittelt werden. Dabei stehen keine Informationen über die internen Strukturen der jeweiligen Agenten zur Verfügung.

Damit kommt dem Erkennen und Verstehen von Verhalten auf Basis von Beobachtungen (Verhaltens-Bestimmung) eine besondere Bedeutung zu. Dabei ist nicht nur die Bestimmung des aktuellen Verhaltens eines Interaktionspartners interessant sondern speziell die Vorhersage des Verhaltens. Für die Vorhersage

kann ein Verhaltens-Modell, welches das Verhalten eines oder mehrerer Agenten unter verschiedenen Gegebenheiten beschreibt, dienen. Die automatische Generierung und Aktualisierung eines solchen Verhaltens-Modells stellt eine besondere Herausforderung dar, da für diese nur Beobachtungen von Interaktionen mit den jeweiligen Agenten zur Verfügung stehen.

Diese Arbeit beschäftigt sich mit dem Modellieren von Agenten-Verhalten und stellt automatisierte Verfahren zur Verhaltens-Bestimmung, zur Generierung von Verhaltens-Modellen und zur Anwendung von Verhaltens-Modellen für die Verhaltens-Vorhersage vor. Um die Tauglichkeit dieser Verfahren zu untersuchen werden sie für ein komplexes Multiagentensystem ausgearbeitet und evaluiert.

Es existiert eine Vielzahl von Multiagentensystemen, die sich hinsichtlich ihrer Komplexität und in vielen Merkmalen unterscheiden [55]. Ein populäres Multiagentensystem steht mit der *RoboCup*-Initiative [28] zur Verfügung. In einem weltweiten Forschungsprojekt beschäftigt sie sich mit der Entwicklung von autonomen Fußballagenten, die unter realitätsnahen Bedingungen gegeneinander Fußball spielen. Durch seine zahlreichen Herausforderungen und interessanten Fragestellungen ist die *RoboCup*-Initiative inzwischen zu einem der weltweit am meisten beachteten Forschungsprojekte der Künstlichen Intelligenz geworden. Die *RoboCup*-Initiative betrachtet dabei sowohl den Roboterfußball als auch das virtuelle Fußballspiel.

Im Rahmen dieser Arbeit wird als Anwendungsszenario das virtuelle Fußballspiel verwendet. Die Multiagentenumgebung wird hierbei durch ein Programm namens *SoccerServer* [46] bereitgestellt. Basierend auf der aktuellen Spielsituation übermittelt der *SoccerServer* den Spieler-Agenten ihre jeweiligen Sichtinformationen, wertet deren Aktionen aus und simuliert die sich daraus ergebenden Bewegungen von Ball und Spieler, um daraus eine aktualisierte Spielsituation zu ermitteln.

Das durch den *SoccerServer* und Spieler-Agenten bereitgestellte Multiagentensystem ist in vieler Hinsicht realistisch. Die Spieler-Agenten müssen in einer hochdynamischen Umgebung mit schnell wechselnden Situationen, auf Basis von begrenzten lokalen Informationen, Entscheidungen in Echtzeit treffen. Dabei ist die Wahrnehmung und die Aktionsausführung fehlerbehaftet und unzuverlässig, die Kommunikation zwischen den Spielern ist eingeschränkt und die Agenten haben begrenzte Kraftreserven.

Um erfolgreich zu sein, müssen die Spieler eines Teams zusammenarbeiten und dabei die Verhalten der gegnerischen Spieler berücksichtigen. Die Bestimmung und Vorhersage von Verhalten gerade der gegnerischen Spieler kann zur Optimierung der Interaktionen mit diesen und damit zur Steigerung der Spielstärke eines Teams beitragen. Für die Verhaltens-Modellierung wird ein Coach-Agent verwendet, der vom *SoccerServer* sehr genaue und weitgehend vollständige Daten über die jeweiligen Spielsituationen erhält.

Obwohl die Optimierung von Agenten-Verhalten basierend auf Verhaltens-Modellen von Interaktionspartnern eine interessante Herausforderung darstellt, beschränkt sich diese Arbeit auf die Bestimmung und die Vorhersage von Verhalten.

Für die Verhaltens-Bestimmung werden in dieser Arbeit Verhalten von einzelnen Spielern bzw. von kleineren Spielergruppen durch Verhaltens-Muster spezifiziert. Basierend auf diesen Verhaltens-Mustern werden Beobachtungen realer Spielverläufe analysiert, um die Verhalten der Spieler zu erkennen.

Für die Verhaltens-Vorhersage wird ein fallbasiertes Verhaltens-Modell generiert, dessen Fälle aus Paaren von erkannten Verhalten und deren Auslösern bestehen. Als Auslöser werden relevante Auszüge der Spielsituation zum Startzeitpunkt des jeweiligen Verhaltens ermittelt. Mit der Verwendung eines Ähnlichkeitsmaßes ist es möglich, Verhalten für Spielsituationen vorherzusagen in denen die Spieler-Agenten bisher nicht beobachtet werden konnten.

Die vorliegende Arbeit gliedert sich in acht Kapitel. Die Kapitel 2 und 3 sind dabei allgemein gehalten, d.h. in ihnen wird von konkreten Anwendungsszenarien abstrahiert. Die Kapitel 4 bis 7 beschreiben den Ansatz der Verhaltens-Bestimmung, der Generierung von Verhaltens-Modellen und der Anwendung von Verhaltens-Modellen am Anwendungsszenario des virtuellen Fußballspiels. Kapitel 8 abstrahiert wieder vom Fußballspiel.

Kapitel 2 beschäftigt sich mit der Zielstellung und der Motivation der Dissertation und setzt sie zu themenverwandten Arbeiten in Bezug. Die Nützlichkeit der Verwendung von Verhaltens-Modellen bei Interaktionen zwischen Agenten wird herausgearbeitet und es wird anhand anderer Veröffentlichungen untersucht, inwieweit die Bestimmung und die Vorhersage von Verhalten bereits angewendet wird.

In Kapitel 3 wird der verwendete Modellierungsansatz vorgestellt. Als Grundlage für die Verhaltens-Modellierung werden Beobachtungen der Umwelt zu diskreten Zeitpunkten eingeführt und nach der genaueren Vorstellung des verwendeten Anwendungsszenarios konkretisiert. Nach Definitionen für Verhalten, Auslöser, Verhaltens-Muster und Auslöser-Muster werden die Prozesse der Generierung und der Anwendung von Verhaltens-Modellen weiter strukturiert und in ihren Grundzügen beschrieben. Die für den vorgestellten Ansatz getroffenen Annahmen werden angegeben.

Kapitel 4 widmet sich dem Prozess der Verhaltens-Bestimmung. Hier werden mehrere Verhaltens-Muster für das virtuelle Fußballspiel spezifiziert. Verhalten sind dabei aus Ereignissen zusammengesetzt, die atomare Beziehungen zwischen Agenten, Objekten und der Umgebung beschreiben. Die Ereignisse, die Verhalten und der Erkennungsprozess werden formalisiert. Desweiteren wird eine objektorientierte Realisierung der Verhaltens-Bestimmung vorgestellt. Dazu werden die Ereignisse eines Verhaltens in verschiedene Gruppen aufgeteilt. Unter der Vorraussetzung, daß sich jedes Verhalten mindestens durch eine Sequenz von Ereignissen definieren lässt, deren Ereignisse sich gegenseitig ablösen, wird ein effizienter Algorithmus zur Erkennung von Verhalten angegeben.

In Kapitel 5 wird der Prozess der Generierung von Verhaltens-Modellen betrachtet. Nach der Spezifikation von Auslöser-Mustern für einige der eingeführten Verhaltens-Muster, werden die Auslöser sowie ihr Erkennungsprozeß formalisiert. Auslöser-Verhaltens-Paare bilden Fälle, die in eine Fallbasis integriert werden. Es wird jeweils ein zusammengesetztes Ähnlichkeitsmaß zwischen Verhalten und zwischen Auslösern spezifiziert und definiert. Teile des

Ähnlichkeitsmaßes für die Auslöser werden automatisch an das Ähnlichkeitsmaß der Verhalten angepaßt, so daß bei ähnlichen Auslösern zweier Fälle auch die dazugehörigen Verhalten ähnlich sind. Eine objektorientierte Realisierung der Generierung von Verhaltens-Modellen wird vorgestellt.

Kapitel 6 erläutert die Anwendung von Verhaltens-Modellen zur Vorhersage von Verhalten. Es wird geklärt, in welchen Spielsituationen Verhalten vorhergesagt werden. Für die entsprechenden Spielsituationen werden die Auslöser bestimmt und als Anfragen an die Fallbasis verwendet. Damit werden die Fälle ermittelt, die den Auslösern am ähnlichsten sind. Basierend auf den ähnlichsten Fällen wird durch Auswahl und Aufbereitung eines Falles das Verhalten vorhergesagt. Für die automatische Aktualisierung des Verhaltens-Modells wird das tatsächlich ausgeführte Verhalten bestimmt und mit der Vorhersage verglichen. Bei fehlerhafter Vorhersage wird sofort ein neuer Fall in die Fallbasis aufgenommen.

In Kapitel 7 wird der vorgestellte Ansatz am Beispiel des virtuellen Fußballspiels evaluiert. Dafür wurden der Ansatz und mehrere ballorientierte Verhaltens-Muster implementiert. Zur Evaluierung wurden die Aufzeichnungen von Spielen eines *RoboCup*-Turniers verwendet. Die Verhaltens-Bestimmung wird dabei für zufällig ausgewählte Beobachtungssequenzen mit menschlichen Beobachtungen verglichen. Die Ergebnisse der Verhaltens-Vorhersage werden dagegen automatisch mit dem tatsächlich ausgeführten Verhalten verglichen.

In Kapitel 8 werden schließlich die wesentlichen Beiträge der Arbeit zusammengefasst. Desweiteren werden Möglichkeiten zur Generalisierung des Ansatzes angegeben und mögliche zukünftige Erweiterungen diskutiert. Für die Generalisierung wird untersucht welche Modifikationen nötig sind um die in den Kapiteln 4 bis 7 für das virtuelle Fußballspiel eingeführten Konzepte auch für andere Anwendungsszenarien zu verwenden.

Für die Darstellung der Konzepte in Analyse, Entwurf und Implementierung werden u.a. Diagramme der UML (Unified Modelling Language) verwendet. In Anhang A wird ein Überblick über die genutzten Diagramme gegeben. Da bei der Entwicklung die englische Sprache verwendet wurde, sind auch die meisten Diagramme und Abbildungen in dieser Sprache abgefasst. Falls die Bedeutung englischer Begriffe nicht aus dem Zusammenhang ersichtlich wird, ist eine deutsche Übersetzung angegeben.

Im Rahmen der Dissertation ist die Diplomarbeit von Uwe Müller [43] entstanden, in der der Prozess der Verhaltens-Bestimmung betrachtet wird. Diese Diplomarbeit stellt eine alternative Realisierung zur Spezifikation und zum automatischen Erkennen von Verhalten dar, die aber im Ansatz auf den gleichen Ideen beruht.

Kapitel 2

Motivation und Einordnung

In diesem Kapitel wird das Anliegen der Dissertation motiviert und die Dissertation zu themenverwandten Arbeiten in Bezug gesetzt. Es wird mit einer allgemeinen Zielstellung begonnen (Abschnitt 2.1). Nach einer Begründung des Bedarfs an Verhaltens-Modellen bei Interaktionen zwischen Agenten (Abschnitt 2.2), wird anhand verwandter Arbeiten die Generierung, Aktualisierung und Anwendung von Verhaltens-Modellen betrachtet (Abschnitt 2.3.2). Dazu untersuchen wir, wie Verhalten automatisch erkannt und verstanden werden können, stellen unterschiedliche Merkmale von Verhaltens-Modellen dar, analysieren Anwendungsmöglichkeiten dieser Modelle und demonstrieren kurz wie diese automatisch generiert werden können. Das Kapitel schließt mit einer Einordnung dieser Arbeit ab (Abschnitt 2.4).

2.1 Zielstellung

Menschen interagieren mit ihren Mitmenschen auf vielfältige Art und Weise. Sei es, daß sie gemeinsam Aufgaben lösen, Konflikte bewältigen oder in Konkurrenz stehen. Bei solchen Interaktionen stellt sich ein Beobachter oft die Frage, was spezielle andere Menschen – die Interaktionspartner (IP) – gerade denken. Dabei sind im allgemeinen nur die Gedanken und die daraus resultierenden Handlungen interessant, die Auswirkungen auf den Beobachter haben.

In Abbildung 2.1 auf der nächsten Seite ist ein Beobachter zu sehen, der sich verschiedene Fragen über einen seiner IP stellt. Ausgehend von der allgemeinen Frage was der IP denkt, kann ein Beobachter daran interessiert sein was sein IP für Wissen hat, was er gerade macht, was er zu tun beabsichtigt, was für Fähigkeiten er hat und wie er auf mögliche Aktionen des Beobachters reagiert.

Ein aufmerksamer und intelligenter menschlicher Beobachter kann diese Fragen bis zu einem gewissen Grad beantworten. In dieser Arbeit wird untersucht, inwieweit dazu auch ein autonomer Agent in der Lage ist. Dabei konzentrieren wir uns auf die beiden Fragen:

- Was macht der IP gerade?
- Was beabsichtigt der IP zu tun?

Was macht der IP gerade? Eine Grundvoraussetzung für das Verständnis eines IP, ist das **Erkennen** und die **Interpretation** seiner Handlungen.

Interaktions-
partner (IP)

Beobachter

Was denkt er?

Was für Wissen hat er?

Was macht er gerade?

Was beabsichtigt er zu tun?

Was für Fähigkeiten hat er?

Wie wird er auf meine Aktionen reagieren?

Abbildung 2.1: Fragen eines Beobachters über einen seiner Interaktionspartner

Die Aktionsfolgen des IP müssen dabei auf geeignete Weise verallgemeinert werden, so daß eine möglichst abstrakte Handlungsbeschreibung entsteht, die auch für einen Menschen transparent ist.

Das Kapitel 4 widmet sich der automatischen Erkennung und Interpretation der Handlungen von Interaktionspartnern.

Was beabsichtigt der IP zu tun? Die **Vorhersage** von Handlungen eines IP ähnelt auf dem ersten Blick einem Orakelspruch. Als Grundlage zur Beantwortung dieser Frage werden sowohl Annahmen über das Wissen des IP, als auch die in der Vergangenheit erkannten Handlungen des IP genutzt. Dabei wird in dieser Arbeit davon ausgegangen, daß der IP bei gleichen oder ähnlichen Gegebenheiten auch gleiche bzw. ähnliche Handlungen ausführt.

In den Kapiteln 5 und 6 wird gezeigt, wie ein Agent die Handlungen seiner IP vorhersagen kann.

Das automatische Erkennen und die Interpretation sowie die Vorhersage von Handlungen der IP ist für viele Anwendungsszenarien von großer Bedeutung, z.B.:

- bei Mannschaftsspielen, wie z.B. Fußball [30],
- bei der Interaktion von Robotern im Katastrophen-Einsatz [29, 45],
- bei der Analyse von biologischen Systemen, wie z.B. Ameisenstaaten [2],
- im Straßenverkehr mit autonomen Fahrzeugen [24, 47],
- in militärischen Szenarien, wie z.B. bei der Unterstützung des Luftkampfes [50, 56, 58].

Die ersten drei dieser Anwendungsszenarien werden näher vorgestellt.

Mannschaftsspiele: Bei Mannschaftsspielen, wie z.B. dem virtuellen Fußballspiel [30] treffen zwei Teams mit entgegengesetzten Zielen aufeinander. Die Strategie eines Teams hat maßgeblich Einfluß darauf, ob das Team erfolgreich ist oder nicht. Die Effektivität einer Strategie hängt dabei von dem Verhalten des gegnerischen Teams ab und sollte daher an das jeweilige gegnerische Team angepaßt werden. Dafür ist es vorteilhaft, die Handlungen der IP des Gegners erkennen, interpretieren und vorhersagen zu können.

Roboter im Katastrophen-Einsatz: Es ist geplant, Roboter verstärkt bei der Suche nach verschütteten Menschen in eingestürzten Gebäuden einzusetzen [29, 45]. Dabei besteht Bedarf an hochspezialisierten Robotern wie Kitano u.a. in [29] darlegen. Diese Roboter müssen für eine weitgehend vollständige Suche in den Trümmern eng zusammenarbeiten. Dafür ist neben Kommunikation, die gerade in Katastrophengebieten ausfallen kann, das Erkennen und die Interpretation von Handlungen anderer Roboter nützlich.

Analyse biologischer Systeme: Bei der Beobachtung und der Analyse von biologischen Systemen, wie z.B. Ameisenstaaten können nicht nur neue biologische Erkenntnisse gewonnen werden, sondern auch Multi-Agenten-Techniken in realen Systemen angewendet werden [2]. Bei Kolonien mit über 100 Ameisen kommt der Zeit-effizienten Erkennung und Interpretation von Ameisen-Verhalten eine besondere Bedeutung zu. Erkenntnisse über das Verhalten von Ameisen können als Grundlage für die Verbesserung von „Ameisen-Algorithmen“[1] genutzt werden.

2.2 Kooperation und Konkurrenz zwischen Agenten

In diesem Abschnitt wird der Bedarf an kommunikationsloser Kooperation bzw. Konkurrenz herausgearbeitet und der Einsatz von Verhaltens-Modellen motiviert. Dazu werden zuerst die Begriffe Agent, Interaktionspartner und Beobachter definiert (Abschnitt 2.2.1). Im folgenden werden die Konzepte von Kooperation und Konkurrenz vorgestellt und Einschränkungen von Kommunikation erarbeitet (Abschnitt 2.2.2). Schließlich werden einige Ansätze bzgl. kommunikationsloser Kooperation vorgestellt (Abschnitt 2.2.3).

2.2.1 Agenten, Interaktionspartner und Beobachter

Der Begriff Agent ist in der Literatur nicht eindeutig bestimmt, er wird in den Arbeiten verschiedener Autoren (z.B. [18, 63]) unterschiedlich definiert. Die in dieser Arbeit verwendete Agenten-Definition orientiert sich an der von Franklin und Graesser [18].

[1] „Ameisen-Algorithmen“ werden beim Routing in Netzwerken, bei der Roboter-Navigation und bei Scheduling-Problemen eingesetzt [12].

Definition 2.1 (Agent) *Ein Agent ist eine autonome Einheit mit Zielen oder Aufgaben, welche über einen längeren Zeitraum in einer Umgebung interagiert. Der Agent nimmt die Umgebung oder Teile dieser wahr, verarbeitet diese Informationen und führt entsprechend seiner Ziele bzw. Aufgaben, Aktionen in der Umgebung aus. Der Agent ist dabei Teil der Umgebung.*

Wir sprechen von einem Multi-Agenten-System (MAS), wenn sich mehr als nur ein Agent in einer Umgebung befinden. In dieser Arbeit werden komplexe MAS, die sich an den Gegebenheiten der Alltagswelt orientieren, betrachtet. Solche **komplexen MAS** haben

- einen sehr großen diskreten oder sogar kontinuierlichen Zustands- und Aktionsraum,
- die Wahrnehmung der Agenten ist unvollständig und fehlerbehaftet,
- die Aktionsausführung der Agenten ist unzuverlässig und
- die Agenten haben infolge der dynamischen Umwelt nur eine begrenzte Zeit um zu Entscheidungen zu gelangen.

Beobachter und Interaktionspartner sind spezielle Agenten, die durch Beziehungen zu anderen Agenten gegeben sind. Wie schon angedeutet, nehmen wir die **Perspektive eines Beobachters** ein, d.h. diese Arbeit stellt einen Ansatz zur Lösung der Probleme eines autonomen Beobachters vor. Alle weiteren Betrachtungen werden aus Sicht des Beobachters durchgeführt.

Definition 2.2 (Interaktionspartner (IP)) *Ein Interaktionspartner (engl. interaction partner) ist ein Agent in einem MAS, dessen Handlungen Auswirkungen auf die Erfüllung der Ziele anderer Agenten haben.*

Führt ein Agent also Aktionen aus, die Auswirkungen auf die Zustandsänderung der Umgebung haben und beeinflußt diese Zustandsänderung die Erfüllung der Ziele anderer Agenten, dann ist dieser Agent ein IP. Die Beeinflussung der Ziele der anderen Agenten muß dabei nicht unmittelbar erfolgen sondern kann sich auch verspätet ergeben.

Definition 2.3 (Beobachter) *Ein Beobachter (engl. observer) ist ein Agent der zielgerichtet die Aktionen seiner Interaktionspartner und/oder deren Auswirkungen beobachtet, um Wissen über diese zu erwerben und/oder zu nutzen.*

Obwohl jeder Agent seine Umgebung und damit seine IP beobachtet, wird mit dem Begriff Beobachter eine spezielle Sorte von Agenten eingeführt um die weiteren Ausführungen verständlicher zu machen. Wissen im Sinne der Definition können z.B. Ansichten, Pläne, Absichten oder Verhalten der IP sein.

Einem Beobachter ist es meist nicht möglich alle Aktionen seiner Interaktionspartner zu beobachten, sei es weil einige Aktionen nicht direkt beobachtbar sind oder weil sie außerhalb des Wahrnehmungsbereiches des Beobachters liegen. Solche Aktionen der IP haben aber Auswirkungen im MAS, die manchmal von einem Beobachter wahrgenommen werden können.

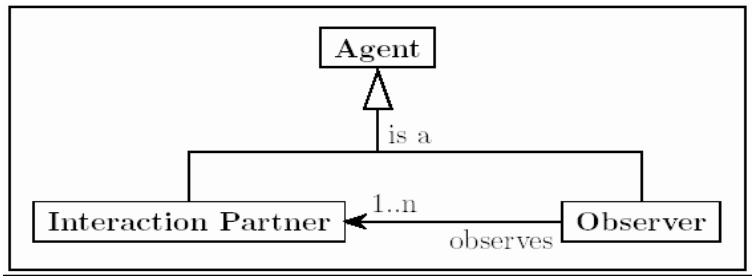


Abbildung 2.2: Zusammenhänge: Agent, Interaktionspartner und Beobachter

Abbildung 2.2 verdeutlicht die Beziehungen zwischen Agenten, Interaktionspartnern und Beobachtern. Ein Beobachter kann auch selbst ein IP für einen anderen Beobachter sein. Dieser Aspekt wird in dieser Arbeit nicht verfolgt.

2.2.2 Kooperation, Konkurrenz und Kommunikation

Das Bilden von Gruppen oder Teams von Agenten in MAS hat den Zweck, gemeinsame oder ähnliche Ziele besser verfolgen zu können. Um diese Ziele zu erreichen, müssen die Agenten eines Teams miteinander kooperieren und gegebenenfalls dabei auftretende Konflikte lösen. Je besser die Agenten kooperieren, um so effizienter kann das Team seine Ziele verfolgen. Einige Aufgaben sind sogar nur von einem Team vollbringbar, was daran liegt, daß die zur Lösung der Aufgabe benötigten Ressourcen bzw. Fähigkeiten im Team verteilt sind. Einer einzelnen Ameise wird es z.B. nicht gelingen, ein größeres Insekt zu erlegen und in den Bau zu bringen. Für eine koordinierte Gruppe von Ameisen ist dies dagegen eine Routine-Aufgabe.

Desweiteren treffen auch Agenten oder Agententeams mit entgegengesetzten oder nicht zu vereinbarenden Zielen aufeinander. Hier müssen die konkurrierenden Agenten oder Teams die auftretenden Konkurrenz-Situationen entsprechend behandeln, um ihre jeweiligen Ziele erfüllen zu können. Bei Teamspielen wie z.B. Fußball sind die Ziele beider Teams konträr, d.h. die Erfüllung der Ziele des einen Teams bedeutet die Nicht-Erfüllung der Ziele des anderen Teams.

Das klassische Mittel für eine gute Kooperation zwischen Agenten ist Kommunikation. Mit Hilfe einer gemeinsamen Kommunikationssprache können sich Agenten vor gemeinsamen Handlungen miteinander abstimmen.

In Konkurrenz-Situationen kann Kommunikation auch ein gutes Mittel zur Beseitigung von Konflikten sein. So ist es prinzipiell möglich durch Absprachen und Verhandlungen Konflikte aufzulösen. Meist sind in Konkurrenz-Situationen die Ziele starr vorgegeben[2], so daß die Agenten der Teams nicht gewillt oder in der Lage sind, über ihre Ziele zu verhandeln. So ist es nur schwer vorstellbar, daß die Teams zweier Fußballmannschaften über den Ausgang eines Fußballspiels miteinander verhandeln und zu einem einvernehmlichen Ergebnis gelangen. Damit ist der Einsatz von Kommunikation in Konkurrenz-Situationen in diesem Fall sehr begrenzt.

[2]Beim Fußball besteht das Ziel eines Teams darin, mehr Tore als das gegnerische Team zu schießen. Bei militärischen Szenarien ist das Ziel durch das Missionsziel vorgegeben.

Auch in kooperativen Szenarien kann Kommunikation Einschränkungen unterliegen. Die Grundvoraussetzung für Kommunikation ist die Fähigkeit, eine gemeinsame Sprache zu sprechen und zu verstehen. Desweiteren kann Kommunikation aufwendig, eingeschränkt oder sogar (zeitweise) unmöglich sein. So ist es für die Spieler beim Fußballspiel nicht sinnvoll, jede Aktion mit ihren Mitspielern abzusprechen, da eine solche Kommunikation zeitlich gesehen aufwendig ist. Das Kommunikationssystem eines Roboters kann ausfallen oder gestört sein. Manchmal ist es auch wichtig Kommunikation gegenüber Dritten geheim zu halten. Es besteht also auch Bedarf an Kooperation ohne Kommunikation.

2.2.3 Kooperation ohne Kommunikation

Genesereth u.a. [20] waren die ersten Autoren, die sich mit dem Thema der Kooperation ohne Kommunikation intensiver auseinandersetzten. In ihrer Arbeit untersuchten sie Kooperationsstrategien für 2-Agenten Matrixspiele[3] mit dem Ziel, den Gewinn des eigenen Agenten zu maximieren. Dazu betrachteten die Autoren verschiedene Annahmen über die Aktionsauswahl des anderen Agenten. Sie entwickelten statische Modelle des Verhaltens des anderen Agenten, die auf Rationalitätskriterien beruhen. Eines der Modelle trifft sogar die Annahme, daß sich der andere Agent genauso wie der eigene Agent verhält. Auch in komplexen MAS wird zuweilen die Annahme getroffen, daß sich andere Agenten genauso oder ähnlich wie der betrachtete Agent verhalten (z.B. in [25]).

In dem Szenario der 2-Agenten Matrixspiele tritt der andere Agent in die Rolle des Interaktionspartners und der eigene Agent in die Rolle des Beobachters. Die für dieses Szenario vorgestellten einfachen statischen Modelle sind nicht geeignet, Handlungen von IP in komplexen MAS zu erkennen oder vorherzusagen. Ein modellbasierter Ansatz ist aber prinzipiell auch in komplexen MAS zur Vorhersage von Verhalten der IP geeignet. Damit kann ein Modell über das Verhalten der IP als Grundlage für die Bewältigung von Kooperations- und Konkurrenz-Szenarien in komplexen MAS dienen. Die Modellierung von Verhalten in komplexen MAS ist Gegenstand dieser Arbeit.

Neben den in der Arbeit betrachteten MAS werden auch bei Robotersystemen Kooperationsansätze verfolgt, die keine oder nur geringer Kommunikation bedürfen. In der Arbeit von Kuniyoshi u.a. [34] beruht dieser Ansatz auf einer Beachtungs-Struktur (engl. attentional structure) die u.a. vorgibt, welche Roboter sich an welche anderen Roboter anzupassen haben. Mit der Beachtungs-Struktur klassifizieren die Autoren unterschiedliche Koordinationsprobleme. Bei den näher betrachteten Koordinationsproblemen unterstützt ein beobachtender Roboter einen anderen Roboter (IP) bei der Erledigung sehr konkreter Aufgaben. Bei einer dieser Aufgaben erkennt der Beobachter Hindernisse in der Bewegungsrichtung seines IP und entfernt diese bevor der IP mit diesen kollidiert. Der beobachtende Roboter erkennt, ob und wann er den IP unterstützen muß und verrichtet dann seinen Teil der Aufgabe. Für die Erledigung der Aufgaben trifft der Beobachter implizit Annahmen über das Verhalten der IP.

[3] Ein bekanntes Beispiel für 2-Agenten Matrixspiele ist das Gefangenendilemma.

2.3 Verhaltens-Modellierung

Die Verhaltens-Modellierung beschäftigt sich mit der Generierung, Anwendung und Aktualisierung von Verhaltens-Modellen für Interaktionspartner. Die Verhaltens-Modellierung gehört damit zum Gebiet der Agenten-Modellierung [27], in welchem Verfahren untersucht werden, die einem Agenten erlauben, Wissen über andere Agenten zu erwerben, zu aktualisieren und abzuleiten.

Bei der Verhaltens-Modellierung wird davon ausgegangen, daß das von den IP ausgeführte Verhalten eine oder mehrere Ursachen[4] hat, die durch einen Beobachter im Vorfeld der Verhaltensausführung ausreichend genau und ausreichend vollständig wahrgenommen werden können. Ein Verhaltens-Modell basiert demnach auf dem **Ursache-Wirkungs-Prinzip**, wobei als Wirkung das Verhalten der IP angesehen wird. Die Abhängigkeit zwischen Ursache und Verhalten wird im Verhaltens-Modell durch eine Abbildung wiedergegeben.

Definition 2.4 (Verhaltens-Modell) *Ein Verhaltens-Modell ist eine Struktur* $\mathfrak{M} = [\mathfrak{U}, \mathfrak{V}, \mathfrak{f}]$*, wobei* $\mathfrak{U}$ *eine nichtleere Menge von Ursachen,* $\mathfrak{V}$ *eine nichtleere Menge von Verhalten und* $\mathfrak{f}$ *eine Abbildung* $\mathfrak{f} : \mathfrak{U} \rightarrow \mathfrak{V}$ *ist.*

Als ein Verhalten wird vorerst eine zeitlich begrenzte Abfolge von Agenten-Aktionen verstanden, die auf ein gemeinsames Teilziel ausgerichtet sind. Zwei in der Literatur verwendete Ansätze für die Betrachtung der Ursache von Verhalten werden in Abschnitt 2.3.2 vorgestellt. Die Abbildung $\mathfrak{f}$ kann z.B. durch ein Neuronales Netz, einen Entscheidungsbaum, eine Menge von Regeln oder durch ein Fallbasiertes System gegeben sein.

Verhaltens-Modelle können als Basis für Interaktionen mit anderen Agenten dienen. Mit ihnen ist es möglich, Vorhersagen über die Handlungen von IP zu machen. Eine Grundlage zur automatischen Generierung von Verhaltens-Modellen ist das automatische Erkennen und Interpretieren von Verhalten $v \in \mathfrak{V}$ (Abschnitt 2.3.1). Die Verhaltens-Modelle können verschiedenen Merkmale haben zu denen die verwendete Ursache $\mathfrak{U}$ gehört (Abschnitt 2.3.2). Desweiteren werden verschiedene Anwendungsmöglichkeiten von Verhaltens-Modellen vorgestellt. (Abschnitt 2.3.3). Schließlich wird die automatische Generierung und Aktualisierung von Verhaltens-Modellen $\mathfrak{M}$ betrachtet und im speziellen die Generierung der Abbildung $\mathfrak{f}$ (Abschnitt 2.3.4).

Die Anwendung sowie die automatische Generierung und Aktualisierung eines Verhaltens-Modells wird durch einen speziellen Beobachter, einen Modellierenden Agenten (MA) vorgenommen.

Definition 2.5 (Modellierender Agent (MA)) *Ein Modellierender Agent (engl. modeling agent) ist ein Beobachter, der ein Verhaltens-Modell ausgewählter Interaktionspartner zur Vorhersage ihrer Verhalten anwendet und/oder ein Verhaltens-Modell ausgewählter IP generiert und/oder aktualisiert.*

In Abbildung 2.3 auf der nächsten Seite sind die Beziehungen zwischen Agenten, Interaktionspartnern, Beobachtern und dem Modellierenden Agenten dargestellt. Der MA modelliert im allgemeinen nur einen Teil seiner IP ($m \leq n$). Ein MA kann genauso wie ein Beobachter selber ein IP für andere MA sein.

[4] Ursache im Sinne eines Ausgangszustandes

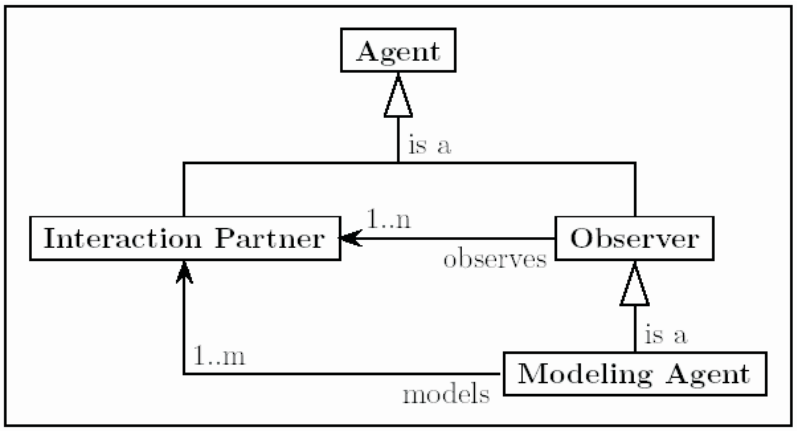


Abbildung 2.3: Zusammenhänge: Agent, Interaktionspartner, Beobachter und Modellierender Agent

2.3.1 Erkennen und Interpretieren von Verhalten

Für die Generierung und Anwendung von Verhaltens-Modellen ist die **Verhaltens-Bestimmung**, also das Erkennen und Interpretieren von Verhalten der IP, eine wichtige Voraussetzung. Die erkannten Verhalten sollen dabei so beschrieben sein, daß sie sich für eine maschinelle Weiterverarbeitung eignen und ebenso für einen Menschen leicht verständlich sind.

In einfachen MAS, wie z.B. rundenbasierten 2-Agenten Matrix-Spielen kann man eigentlich nicht von Verhalten sprechen[5], da die Agenten in jedem abgeschlossenen Szenario einer Runde nur jeweils eine Aktion ausführen. In komplexen MAS setzt sich ein Verhalten im allgemeinen aus mehreren Aktionen eines oder mehrerer IP zusammen. Ein Überhol-Verhalten im Straßenverkehr setzt sich z.B. aus einem Anzeigen des Spurwechsels, einem Beschleunigen, dem Wechsel in die andere Spur, dem eigentlichen Überholen, dem Wechsel in die ursprüngliche Spur und eventuell einem Abbremsen zusammen.

Es werden Verhalten betrachtet, die sich an potenziellen Absichten bzw. Teilzielen der IP orientieren. Ein Verhalten kann als die Umsetzung einer Absicht eines oder mehrerer IP im MAS angesehen werden. Bei einer solchen Auslegung wird in dieser Arbeit von der Annahme ausgegangen, daß die IP ihre Aktionen zielorientiert ausführen und daß sich die Auswirkungen dieser Aktionen im MAS als ein Verhalten beschreiben lassen.

Die IP können Absichten haben, bei denen eine Abbildung auf Verhalten nicht gelingt (MA kennt das Verhalten nicht, oder die Umsetzung der Absicht ist nicht durch den MA beobachtbar). Genauso kann es vorkommen, daß der MA Verhalten erkennt, die keiner Absicht der IP entsprechen (Absicht nicht bewußt oder gar nicht vom IP getroffen). In letzterem Falle spricht man auch von *emergent behavior* (dt. sich automatisch ergebendes Verhalten) (z.B. [61]).

Verhalten können von unterschiedlichster Komplexität sein, einfache Ver-

[5] Die Möglichkeit, die Aktionen eines Agenten über mehrere Runden als Verhalten anzusehen wird hier nicht verfolgt, da mit jeder neuen Runde die Voraussetzung für beide Agenten neu initialisiert wird.

halten bestehen nur aus wenigen Aktionen eines IP, sehr komplexe Verhalten können aus diversen Aktionen mehrerer IP bestehen, wobei einige oder alle der IP nur für einen begrenzten Zeitraum am Verhalten beteiligt sind. Die Komplexität der Verhaltens-Bestimmung ist damit maßgeblich von den betrachteten Verhalten und von den Eigenschaften des MAS abhängig.

Neben der Anwendung der Verhaltens-Bestimmung für die Generierung und Anwendung von Verhaltens-Modellen, kann sie auch für automatische Analysesysteme und Kommentatoren benutzt werden. Die im folgenden betrachteten Arbeiten erkennen und interpretieren Verhalten zu diesem Zwecke. In einigen dieser Arbeiten ist auch genauer ausgeführt, wie die Verhaltens-Bestimmung durchgeführt wird.

Kommentatoren nehmen Verhalten wahr um mitzuteilen, was in der Umgebung die sie beobachten, gerade passiert. Ähnlich einem menschlichen Reporter verwenden sie für ihre Kommentare die natürliche Sprache. André u.a. [1] sowie Voelz u.a. [59] haben den automatischen Kommentator Rocco entwickelt. Rocco nimmt einfache Ereignisse in einer virtuellen Fußballumgebung (*SoccerServer* [46]) wahr und setzt sie zu komplexeren Ereignissen (Verhalten) zusammen.

Ein weiterer automatischer Kommentator für den *SoccerServer* ist Mike, der von Matsubara u.a. [40] entwickelt wurde. Neben der Erkennung von einfachen Verhalten analysiert dieses System auch einige Aspekte des Verhaltens der IP mit Hilfe von Statistiken. Basierend auf diesen Statistiken trifft Mike sogar Prognosen über den Ausgang von Spielen. Das Analysesystem von Mike wurde als konfigurierbarer Assistent von Frank u.a. [17] zur Verfügung gestellt.

Mit dem Assistenten ISAAC stellen Raines u.a. [49] ein Analysesystem für den *SoccerServer* vor. Dieses System analysiert Aufzeichnungen von Fußballspielen auf verschiedenen Ebenen. So werden Verhaltensweisen einzelner Agenten, Agenten-Interaktionen und das Team-Verhalten analysiert, statistisch ausgewertet und dem menschlichen Nutzer präsentiert. Eine sogenannte Störungsanalyse (engl. perturbation analysis) wird für das Torschuß-Verhalten durchgeführt um dem Nutzer Verbesserungsvorschläge zur Änderung des Agenten-Verhaltens aufzuzeigen. Dazu werden die statistisch stärksten Einflußgrössen für die fehlgeschlagenen Torschüsse eines Spieles bestimmt. Ebenso wie Mike macht ISAAC, basierend auf einfachen Spiel-Statistiken, Prognosen über den Ausgang von Spielen.

Miene und Visser [41] präsentieren einen universellen Ansatz zur Spezifikation und zum Erkennen von Verhalten. Die Verhalten werden durch räumliche und zeitlichen Relationen spezifiziert. Diese Arbeit ist noch in einem frühen Stadium, die automatische Erkennung von Verhalten wird nur sehr eingeschränkt durchgeführt.

Der Erkennung von Spielzügen beim Football widmen sich Intille und Bobick [23]. Als Basis zur Erkennung werden die aufgezeichneten Bewegungen (Trajektorien) der angreifenden Mannschaft genutzt. Aus diesen Bewegungen werden einfache Ereignisse extrahiert. Diese werden in einem Bayesschen Netz (engl. Bayesian Belief Network) genutzt, um das Verhalten jedes Spielers zu erkennen. Die erkannten Verhalten werden mit Hilfe von zeitlichen Einschränkungen und Mustern für Teamspiele kombiniert um das Verhalten des Teams zu interpretieren. Diese Kombination geschieht wiederum durch ein Bayessches

Netz. So kann für jeden Spielzug ermittelt werden, mit welchen Wahrscheinlichkeiten dieser den vom Designer erstellten Teamspiel-Mustern entspricht.

Zusammenfassung der Arbeiten zur Verhaltens-Bestimmung: Alle diese Systeme beschränken sich bei der Verhaltens-Erkennung darauf, **was** für ein Verhalten von den IP ausgeführt wird. **Wie** die IP Verhalten ausführen wird nicht untersucht. Es werden keine Verhaltens-Eigenschaften oder Untergruppen der Verhalten betrachtet[6]. Lediglich in [49] wird eine Unterteilung in erfolgreiches und erfolgloses Verhalten vorgenommen. Damit eignen sich die vorgestellten Systeme nur wenig als Ausgangspunkt für die Vorhersage von Verhalten, da nur die Existenz eines Verhaltens erkannt wird und nicht dessen Eigenschaften.

Ergebnisse hinsichtlich der Vollständigkeit und Korrektheit der Verhaltens-Erkennung werden in den vorgestellten Arbeiten (außer in [23]) nicht angegeben. Gründe dafür liegen wahrscheinlich in der Schwierigkeit die Korrektheit zu evaluieren, da sie subjektiven Einschätzungen unterliegt und in der nur eingeschränkten Automatisierbarkeit des Evaluierungsprozesses.

2.3.2 Merkmale von Verhaltens-Modellen

In diesem Abschnitt werden Merkmale und Eigenschaften der Verhaltens-Modellierung und von Verhaltens-Modellen betrachtet. Zuerst werden Anmerkungen zur Existenz von Verhaltens-Modellen gemacht (Abschnitt 2.3.2.1) und untersucht woher Verhaltens-Modelle kommen können (Abschnitt 2.3.2.2). Eigenschaften der Abbildung $\mathfrak{f}$ (Abschnitt 2.3.2.3) und der Verhalten $\mathfrak{V}$ (Abschnitt 2.3.2.4) werden im folgenden analysiert. Schließlich werden mit aktionsbasierten Verhaltens-Modellen (Abschnitt 2.3.2.5) und situationsbasierten Verhaltens-Modellen (Abschnitt 2.3.2.6) zwei in der Literatur verwendete Ansätze für die Betrachtung der Ursache $\mathfrak{U}$ von Verhalten vorgestellt. Die Einordnung von verwandten Arbeiten bzgl. der Merkmale wird nach deren genauerer Analyse angegeben (Abschnitt 2.3.3.6).

2.3.2.1 Existenz von Verhaltens-Modellen

Die Modellierung von Verhalten kann **implizit** oder **explizit** erfolgen. Bei der impliziten Verhaltens-Modellierung werden die Handlungen der IP nur indirekt berücksichtigt, d.h. ein Verhaltens-Modell $\mathfrak{M}$, genauer dessen Abbildung $\mathfrak{f}$, ist nicht identifizierbar. Bei der expliziten Verhaltens-Modellierung dagegen ist ein Verhaltens-Model $\mathfrak{M}$ vorhanden.

Implizite Verhaltens-Modellierung kann bei der Verwendung von Lernverfahren (z.B. Neuronale Netze oder Entscheidungsbäume) für die Erstellung bzw. Verbesserung der Entscheidungsroutinen eines Agenten in Szenarien mit IP auftreten, wenn die Aktionen der IP nicht direkt berücksichtigt werden. Wenn z.B. ein Fußball spielender Agent den Torschuß trainiert, wird indirekt

[6] Durch die Spezifikation vieler hoch spezialisierter Verhalten kann zwar auf eine Unterteilung in Untergruppen und vielleicht auch auf Verhaltens-Eigenschaften verzichtet werden, eine solche massive Spezifikation ist aber sehr aufwendig, wenn überhaupt handhabbar und wird in keiner der untersuchten Arbeiten durchgeführt.

das Verhalten des verwendeten Torwarts bei Torschüssen berücksichtigt. Trifft der Agent auf einen Torwart mit einem abweichenden Modell bei Torschüssen kann das gelernte Torschuß-Verhalten inadäquat sein.

Eine Anpassung auf neue IP ist bei der impliziten Verhaltens-Modellierung aufwendig, da dafür die Entscheidungsroutinen angepaßt werden müssen. Infolge des Fehlens eines Verhaltens-Modells sind die Handlungen der IP nicht transparent und eignen sich deshalb nicht zur Verhaltens-Vorhersage. Damit ist die implizite Verhaltens-Modellierung für den Ansatz dieser Arbeit nicht geeignet.

Bei der expliziten Verhaltens-Modellierung kann eine Anpassung auf neue IP bereits durch den Austausch oder die Anpassung eines Verhaltens-Modells vorgenommen werden. Für das obige Beispiel des den Torschuß lernenden Spielers kann auch eine explizite Verhaltens-Modellierung des Torwarts realisiert werden. Dies geschieht, indem die Entscheidung des Schützen, wie der Torschuß auszuführen ist, nicht nur von der aktuellen Situation abhängig gemacht wird, sondern auch von dem vermuteten Verhalten des Torwarts, welches durch ein Verhaltens-Modell gegeben ist.

Verhaltens-Modelle sind zur Vorhersage von Agenten-Handlungen geeignet. Im weiteren Verlauf wird nur die explizite Verhaltens-Modellierung betrachtet, alle im weiteren betrachteten Arbeiten verwenden Verhaltens-Modelle.

2.3.2.2 Woher kommen Verhaltens-Modelle

Verhaltens-Modelle, genauer die Abbildung $\mathfrak{f}$, können **manuell** sowie **automatisch** erstellt werden. Die automatische Generierung von Verhaltens-Modellen basierend auf wiederholten Beobachtungen der IP durch den MA wird in Abschnitt 2.3.4 auf Seite 22 ausführlich betrachtet. Bei der manuellen Erstellung definiert ein **Designer**, bei welchen Ursachen die IP welche Verhalten ausführen.

Unter der Annahme, daß sich der IP genauso oder ähnlich wie der MA verhält können auch die **Entscheidungsroutinen des MA** als Verhaltens-Modell oder als Basis zur Generierung eines Verhaltens-Modells verwendet werden. Bei der direkten Benutzung der Entscheidungsroutinen als Modell für den IP sagt der MA das Verhalten des IP voraus, indem er simuliert, was für Handlungen er bei der gegebenen Ursache ausführen würde. Diese Art Verhaltens-Modell funktioniert in homogenen MAS bzw. innerhalb homogener Teams, in denen alle Agenten die gleichen Entscheidungsroutinen aufweisen. Der Erfolg in heterogenen MAS ist dagegen eher unrealistisch.

Die Agenten des Fußball-Teams *AT Humboldt 98* [21, 44] nutzen z.B. ein einfaches Verhaltens-Modell dieser Art. Bei der Entscheidung, ob und wohin der Ball gepaßt werden soll, nutzt der Agent die eigene Ball-Abfangstrategie als Modell, um abzuschätzen wie schnell seine Mitspieler und seine Gegner den Ball abfangen können.

2.3.2.3 Statische oder adaptive Modelle

Mit der Unterscheidung in **statische** und **adaptive** Modelle wird eine Eigenschaft der Abbildung $\mathfrak{f}$ des Verhaltens-Modells betrachtet. Einmal generierte

statische Verhaltens-Modelle ändern sich nicht mehr. Adaptive Modelle passen sich während der Interaktionen mit den IP an diese an, d.h. die Abbildung $\mathfrak{f}$ wird entsprechend aktualisiert, wenn die Vorhersage des Modells fehlerhaft war.

In Szenarien mit homogenen Agenten leisten statische Verhaltens-Modelle gute Dienste, insofern sie das Verhalten der IP korrekt repräsentieren. In anderen Szenarien repräsentieren statische Verhaltens-Modelle einen durchschnittlich oder optimal agierenden IP (zumindest gilt dies für alle untersuchten Arbeiten). Damit können statische Modelle nur eine Abschätzung des Verhalten der IP geben.

Statische Modelle sind demnach mehr oder weniger auf homogene Agenten-Gruppen begrenzt. Die Verwendung statischer Modelle ist insbesondere fraglich, wenn der MA auf IP trifft, die von anderen Entwicklern erstellt wurden, was in Konkurrenz-Situationen oft der Fall ist.

Mit adaptiven Verhaltens-Modellen ist eine Anpassung an IP möglich, die von anderen Entwicklern erstellt wurden. Sie erlauben auch eine Aktualisierung eines bereits existierenden Modells eines IP, wenn dieser sein Verhalten ändert. Adaptive Verhaltens-Modelle müssen automatisch erstellt werden, d.h. der MA muß seine IP über einen längeren Zeitraum gezielt beobachten und deren Verhalten analysieren. Prinzipiell wird für jeden IP ein eigenes Verhaltens-Modell benötigt.

Es existieren auch Grenzfälle zwischen statischen und adaptiven Modellen. Die Parameterisierung von ansonsten statischen Modellen und die Anpassung an IP durch Auswahl des geeignetsten Modells aus einer Menge von vorgefertigten Verhaltens-Modellen sind zwei solche Grenzfälle. Solche Verhaltens-Modelle werden wir **semi-adaptiv** nennen.

2.3.2.4 Agenten-, Gruppen- oder Team-Modelle

Verhaltens-Modelle können für individuelle **Agenten**, für **Gruppen** oder für **Teams** gegeben sein. Mit dieser Unterteilung wird eine Eigenschaft der untersuchten Verhalten $\mathfrak{V}$ betrachtet.

Bei Agenten-Modellen wird deren Verhalten als isoliert gegenüber anderen Agenten betrachtet, Verhalten, die Interaktionen mit anderen IP darstellen, werden ignoriert oder reduziert ohne Interaktionen betrachtet[7]. Mit Gruppen-Modellen können Verhalten betrachtet werden, an deren Ausführung mehr als ein IP beteiligt ist. Die Gruppen werden dabei dynamisch gebildet, wobei jeder teilnehmende IP eine oder mehrere Rollen bei einem Gruppen-Verhalten einnimmt. Team-Modelle beschreiben das Verhalten eines ganzen Teams indem sie die verfügbaren Rollen des Teams unter den Agenten des Teams aufteilen. Im Gegensatz zu Gruppen-Modellen, bei denen nur ausgewählten Agenten eines Teams Rollen zugewiesen werden, muß bei Team-Modellen jedem Agenten eine Rolle zugewiesen werden.

Agenten-, Gruppen- und Team-Modelle können auch in einem Verhaltens-Modell kombiniert werden. Unter der Annahme, daß die zu modellierenden IP eines Teams homogen sind wird nur ein Verhaltens-Modell pro Team benötigt.

[7] Die Verwendung eines Agenten-Modells für mehrere Agenten in einem MAS schließt nicht aus, daß Gruppen-Verhalten als Resultat entsteht.

Andernfalls müßte für jeden IP und für jede IP-Kombination für die Gruppen- oder Team-Modelle benötigt werden ein eigenes Verhaltens-Modell generiert werden. Tatsächlich beschränken sich die Autoren aller im folgenden untersuchten Arbeiten auf ein Verhaltens-Modell pro Team.

2.3.2.5 Aktionsbasierte Verhaltens-Modelle

Bei der Verwendung eines aktionsbasierten Verhaltens-Modells werden die ausgeführten Handlungen (Aktionen) der IP als Ursache zur Bestimmung von deren Verhalten oder Absichten genutzt. Unter der Voraussetzung, daß die Aktionen der IP beobachtet oder anders bestimmt werden können, beruht dieser Ansatz auf der Annahme, daß die IP mit den vergangenen und aktuellen Aktionen eine Absicht verfolgen, die aus den Aktionen abgeleitet werden kann und von den IP weiter verfolgt wird.

Gegen die Annahme – daß durch die aktuellen Aktionen der IP deren Absicht bestimmt werden kann[8] – spricht, daß die aktuelle Aktion der Vorläufer für eine große Menge möglicher Zukünfte sein kann [4]. Diese Zukünfte können prinzipiell die Ziele verschiedener Absichten der IP sein. Damit ist die Bestimmung der Absicht eines IP im allgemeinen nicht eindeutig. Mit diesem Ansatz kann also nur eine Menge von möglichen Absichten der IP bestimmt werden.

2.3.2.6 Situationsbasierte Verhaltens-Modelle

Bei der Verwendung eines situationsbasierten Verhaltens-Modells wird jeweils die aktuelle Situation als Ursache der Verhalten der IP verwendet. Diese Herangehensweise beruht auf der Annahme, daß die Handlungen der IP und damit deren beobachtbare Verhalten zu einem Großteil durch deren aktuelle Situationen bestimmt sind.

Diese Annahme wird auch durch einen Kommentar von Bratman gestützt, der in [4] aussagt, daß die Pläne von Agenten konsistent mit deren Ansichten sein sollten. Da die Pläne eines Agenten direkt zu wahrnehmbaren Verhalten führen und die Ansichten eines Agenten zu einem Großteil durch dessen aktuelle Situation bestimmt sind, ist auch eine Konsistenz zwischen den Situationen der IP und deren Verhalten gewährleistet[9].

Da sich die Beobachtungen des MA und der IP im allgemeinen unterscheiden und damit MA und IP verschiedene Ansichten über die Welt haben, stellt die Bestimmung der aktuellen Situation des IP ein erhebliches Problem dar. Eine wichtige Voraussetzung für die Anwendung solcher Verhaltens-Modelle ist also, daß die jeweiligen Situationen der IP von dem MA ausreichend genau wahrgenommen werden können.

[8] Diese Annahme wird in der englischen Literatur auch mit *Behaviorist theory* bezeichnet.

[9] Bei dieser Betrachtungsweise wird auch davon ausgegangen, daß der IP nicht versucht ein falsches Bild von sich zu vermitteln, indem er bewußt Verhalten ausführt um den MA zu täuschen.

2.3.3 Anwendungsmöglichkeiten von Verhaltens-Modellen

In diesem Abschnitt werden verschiedene Anwendungsmöglichkeiten von Verhaltens-Modellen anhand verwandter Literatur untersucht. Dabei wurden auch Arbeiten aufgenommen, die Absichten der IP anstatt Verhalten betrachten.

Die Verwendung von Verhaltens-Modellen für die Vorhersage von Verhalten wird mit der aktionsbasierten (Abschnitt 2.3.3.1) und der situationsbasierten (Abschnitt 2.3.3.2) Verhaltens-Vorhersage untersucht. Die Ergebnisse der Verhaltens-Vorhersage können benutzt werden, um das Verhalten von IP zu überwachen (Abschnitt 2.3.3.3), eine Interaktionsstrategie für den MA zu bestimmen und/oder zu optimieren (Abschnitt 2.3.3.4) und um das Verhalten von IP nachzuahmen (Abschnitt 2.3.3.5). Schließlich werden die in diesem Abschnitt untersuchten Arbeiten hinsichtlich der Merkmale ihrer Verhaltens-Modelle zusammengefasst (Abschnitt 2.3.3.6).

2.3.3.1 Aktionsbasierte Verhaltens-Vorhersage

Tambe und Rosenbloom [56, 58] verfolgen den Ansatz der aktionsbasierten Verhaltens-Vorhersage, den sie als Agenten-Observierung (engl. agent tracking) bezeichnen. Sie lösen die Mehrdeutigkeit der möglichen Absichten von IP auf, indem sie bei gegnerischen IP die Absicht auswählen, die die ungünstigsten Auswirkungen auf den MA haben. In kooperativen Szenarien legt sich der MA auf eine zufällige der möglichen Absichten fest. Bei widersprüchlichen Folge-Beobachtungen wird die vermutete Absicht der IP durch Backtracking (dt. Rückverfolgung) angepaßt. Die Absicht ist dabei durch eine Operator-Hierarchie beschrieben, die von der höchsten Ebene bis zu einzelnen Aktionen der IP reicht. Die Auflösung der Mehrdeutigkeiten und das Backtracking findet auf jeder einzelnen Ebene statt.

Damit repräsentiert das von Tambe und Rosenbloom verwendete Modell in Konkurrenz-Situationen optimal agierende IP und in kooperativen Szenarien randomisierte IP. Der Ansatz wurde unter Echtzeitbedingungen in dynamischen MAS für kooperative IP (Gruppen-Manöver von Hubschraubern) und konkurrierende IP (Luftkampf) angewendet.

Einen ähnlichen Ansatz verfolgen Rao und Murray, die in [50] einen Algorithmus zur Bestimmung von Plänen eines IP vorstellen. Die Pläne sind dabei voneinander unabhängig repräsentiert, so daß der MA gezielt beobachten kann, ob der IP gerade einen bestimmten Plan ausführt oder nicht. Damit hat der MA die Möglichkeit, nur die für die aktuellen Gegebenheiten interessanten Pläne zu überwachen. Auch in der Arbeit von Rao und Murray wird ein statisches Verhaltens-Modell genutzt. Mehrdeutigkeiten werden nicht aufgelöst.

2.3.3.2 Situationsbasierte Verhaltens-Vorhersage

Carmel und Markovitch betrachten in ihren Arbeiten [7, 8] u.a. die Vorhersage von Aktionen für wiederholte 2-Agenten Matrix-Spiele. Diese Arbeit betrachtet auch andere Aspekte von Verhaltens-Modellen und wird daher erst in Abschnitt 2.3.4 auf Seite 22 genauer vorgestellt.

Für komplexe MAS gibt es kaum eine Arbeit die ein situationsbasiertes Verhaltens-Modell zur Vorhersage von Verhalten der IP nutzt. Lediglich Riley und Veloso nutzen in [53] Verhaltens-Modelle zur Vorhersage der Bewegungen der gegnerischen Spieler (die IP) bei Standardsituationen des virtuellen Fußballspiels im *SoccerServer*. Der Aufenthaltsort jedes Spielers des gegnerischen Teams wird als Wahrscheinlichkeitsverteilung modelliert und zur Ermittlung einer optimierten Interaktionsstrategie (siehe Abschnitt 2.3.3.4) für die Spieler des eigenen Teams verwendet.

Der MA generiert dabei das Verhaltens-Modell der IP nicht selber, sondern wählt es aus einer Menge durch den Entwickler vorgegebener Verhaltens-Modelle aus. Es wird das Verhaltens-Modell ausgewählt, welches basierend auf den bisherigen Interaktionen mit den IP die höchste Genauigkeit bei Vorhersagen hatte. Das Verhaltens-Modell liefert für die Ballposition und die Positionen aller Gegner (IP) die zukünftigen Positionen der Gegner als Wahrscheinlichkeitsverteilung. Die Vorhersage von Gegner-Positionen wird zur Bestimmung von Plänen für Standardsituationen, wie z.B. Freistößen angewendet. Ein Coach-Agent (MA) bestimmt die Pläne und übermittelt sie den ausführenden Spieler-Agenten.

2.3.3.3 Verhaltens-Überwachung

Bei der Verhaltens-Überwachung (engl. behavior monitoring) beobachtet der MA seine IP und vergleicht das erkannte Verhalten dahingehend, ob es dem Modell der IP entspricht[10]. Wenn eine Abweichung vom Modell festgestellt wird, ist dies ein Indiz für einen Fehlerfall.

Kaminka und Tambe stellen in [25] ein System namens SAM (Socially Attentive Monitoring) vor, mit welchem ein MA seine IP überwacht. Bei einer Abweichung des Verhaltens des IP von dessen Modell wird durch eine Diagnose überprüft, ob es sich um einen Fehler handelt und gegebenenfalls die Ursache des Fehlers bestimmt. Bei erkannter Ursache unterstützt der MA den IP bei der Korrektur des Fehlers. Die Autoren wenden diesen Ansatz in einer Simulationsumgebung für Gruppen-Manöver von Hubschrauber-Agenten an. Bei den durchgeführten Experimenten nimmt jeweils ein Hubschrauber die Rolle des MA ein. Dieser erkennt Fehler, die auf fehlende Informationen des IP zurückzuführen sind und hilft bei der Korrektur dieser Fehler durch Übermittlung der benötigten Informationen. Die Autoren verwenden beim Verhaltens-Modell die aktuelle Aktion der IP als Ursache.

2.3.3.4 Ermitteln einer Interaktionsstrategie

Ein Ziel bei der Anwendung von Verhaltens-Modellen liegt in der Bewältigung bzw. Unterstützung von Kooperations- und Konkurrenz-Szenarien und damit in der Ermittlung einer an die IP angepaßten Interaktionsstrategie. Mit der Anwendung eines Verhaltens-Modells zur Vorhersage von Verhalten ist es dem MA möglich, sein Verhalten an das der IP anzupassen.

[10] Verhaltens-Überwachung = Vergleich(Verhaltens-Erkennung, Verhaltens-Vorhersage)

Die bisherigen Arbeiten, die basierend auf einem Verhaltens-Modell eine Interaktionsstragie ermitteln, passen sich an gegnerisches Verhalten an, indem sie eine optimierte oder sogar optimale Interaktionsstrategie ermitteln. Carmel und Markovitch demonstrieren die Ermittlung der besten Gegen-Aktion bei 2-Agenten Matrixspielen [7, 8] (genauer in Abschnitt 2.3.4 auf Seite 22 beschrieben). Riley und Veloso stellen eine an die IP angepaßte Ermittlung von Plänen für Standardsituationen vor (genauer in Abschnitt 2.3.3.2 auf der vorherigen Seite beschrieben).

2.3.3.5 Nachahmen von Verhalten

Das Nachahmen von Verhalten ist durch den Wunsch bestimmt, von erfolgreichen IP zu lernen, indem ihr Verhalten imitiert wird. Es bietet sich bei Gegebenheiten an, in denen sich der MA in der gleichen oder ähnlichen Situation befindet in denen er bereits einmal das Verhalten von IP beobachten konnte. Befinden sich IP und MA zeitgleich in der selben Situation, dann ist für das Nachahmen von Verhalten gar kein Verhaltens-Modell nötig. Der MA kann einfach die Aktionen des IP beobachten und sie genauso ausführen wie der IP.

Für ein zeitversetztes Nachahmen ist dagegen ein Verhaltens-Modell nötig mit welchem festgehalten wird, in welchen Situationen die IP bestimmte Verhalten ausführen (es wird das Verhalten und dessen Ursache protokolliert). Das Verhaltens-Modell wird dann nicht zur Vorhersage des Verhaltens der IP angewendet sondern für die Wahl des Verhaltens des MA oder anderer Agenten. Der nachahmende Agent nimmt also einfach seine Situation wahr und führt das Verhalten aus, welches der IP in dieser Situation ausgeführt hätte. Der MA bzw. die anderen nachahmenden Agenten müssen damit auch über entsprechende Fähigkeiten zur Umsetzung des Verhaltens verfügen.

Conte und Paolucci betrachten in [9] das Imitieren von Verhalten als eine Art des sozialen Lernens. In dieser soziologischen Arbeit diskutieren sie verschiedene Aspekte des Nachahmens von Verhalten. Sie motivieren das Nachahmen von Verhalten mit dem Wunsch eines Agenten ähnlich einem IP zu sein oder zumindest ähnlich einem IP zu handeln und zwar solange wie der IP als geeignetes Modell vom nachahmenden Agenten angesehen wird.

Sie erarbeiten notwendige **Voraussetzungen**, um Verhalten von IP nachzuahmen: Neben der Notwendigkeit ein **Verhaltens-Modell** der zu imitierenden IP zu generieren und zu aktualisieren, begründen sie die Notwendigkeit der Betrachtung von **externen Erfolgskriterien**. Mit diesen Erfolgskriterien kann festgestellt werden, ob das Nachahmen eines IP überhaupt sinnvoll oder erfolgsversprechend ist. Ein Fußballspieler z.B. sollte sich an besseren und nicht an schlechteren Fußballspielern beim Nachahmen von Verhalten orientieren. Desweiteren begründen Conte und Paolucci den Bedarf an entsprechenden **Fähigkeiten** zum Imitieren von Verhalten, die eine Grundvorraussetzung für das Nachahmen von Verhalten darstellen. So wird ein zur Fortbewegung nur mit Rädern ausgestattetes Roboter-Fahrzeug beim Nachahmen der Überwindung eines größeren Hindernisses scheitern, wenn es ein ähnlich großes Kettenfahrzeug als Modell nimmt. Das Roboter-Fahrzeug verfügt nicht über die erweiterte Fähigkeit die durch die Ketten des Kettenfahrzeuges gegeben sind.

In [60] wird von Wendler u.a. ein Verhaltens-Modell zum Nachahmen von Agenten-Verhalten angewendet. Für das simulierte Fußballspiel des *SoccerServer* wird u.a. erfolgreiches Paß-Verhalten imitiert. Der Coach beobachtet das Paß-Verhalten des eigenen Teams in mehreren Spielen. Mit den erfolgreichen Päßen generiert der Coach ein eingeschränktes Verhaltens-Modell, mit welchem die Agenten einer zweiten Generation des eigenen Teams ausgerüstet werden. Das Verhaltens-Modell beinhaltet eine Fallbasis, welche aus beobachteten erfolgreichen Pässen besteht. Mit diesem Modell wird das bisherige Paß-Verhalten ersetzt. Die Agenten wenden dieses Verhaltens-Modell für Teile der eigenen Handlungsauswahl an. Die Erstellung des Modells ist dabei nur in Teilen automatisiert, andere Teile des Modells sind durch die Entscheidungsroutinen des nachahmenden Agenten bzw. durch den Designer vorgegeben.

Ein wesentlicher Aspekt von [60] ist die Aufbereitung der einzelnen beobachteten Pässe durch den Coach-Agenten. Dieser modifiziert die Pässe anhand mehrerer theoretischer Koordinationsmodelle bevor er das Verhaltens-Modell an die Spieler-Agenten transferiert.

2.3.3.6 Klassifizierung von themenverwandten Arbeiten

Die Tabelle 2.1 zeigt die Einordnung von themenverwandten Arbeiten bzgl. der Eigenschaften der dort verwendeten Verhaltens-Modelle. Es ist jeweils angegeben woher das Modell kommt sowie welche Eigenschaften die Abbildung $\mathfrak{f}$, das Verhalten $\mathfrak{V}$ und die Ursache $\mathfrak{U}$ haben. Desweiteren ist angegeben, für was das Verhaltens-Modell hauptsächlich verwendet wird. Nur die Arbeit von Carmel und Markovitch [7, 8] beschäftigt sich dabei nicht mit komplexen MAS.

Bei der Angabe von wem die Arbeit stammt, ist in der Tabelle jeweils nur der erste Autor angegeben. Bei der Angabe woher das Modell kommt, bedeutet MA, daß die Entscheidungsroutinen des MA verwendet werden und Designer, daß die Entscheidungsroutinen vom Designer vorgegeben werden. Die Angaben MA und Designer stellen jeweils eine manuelle Generierung des Verhaltens-Modells dar. Obwohl die Angabe **semi-automatisch** impliziert, daß bereits Teile der Modell-Generierung automatisiert sind, ist die Generierung in wesentlichen Teilen dennoch als manuelle Generierung zu verstehen.

Arbeit von	Tambe [56, 58]	Rao [50]	Carmel [7, 8]
Woher Modell	Designer o. MA	Designer	automatisch
$\mathfrak{f}$-Eigenschaft	statisch	statisch	adaptiv
$\mathfrak{V}$-Eigenschaft	Gruppe, Team	Agent	Agent
$\mathfrak{U}$-Eigenschaft	Aktion	Aktion	Situation
Anwendung	Vorhersage	Vorhersage	Strategie

Arbeit von	Riley [53]	Kaminka [25]	Wendler [60]
Woher Modell	Designer	MA	semi-automatisch
$\mathfrak{f}$-Eigenschaft	semi-adaptiv	statisch	statisch
$\mathfrak{V}$-Eigenschaft	Team	Agent	Agent, Gruppe
$\mathfrak{U}$-Eigenschaft	Situation	Aktion	Situation
Anwendung	Strategie	Überwachung	Nachahmen

Tabelle 2.1: Eigenschaften der Verhaltens-Modelle von verwandten Arbeiten

2.3.4 Automatische Generierung von Verhaltens-Modellen

Bisher gibt es keine Systeme, die eine automatische Generierung von adaptiven Verhaltens-Modellen für komplexe MAS durchführen. Alle bisherigen Arbeiten beschränken sich bei komplexen MAS auf eine Vorgabe von Verhaltens-Modellen durch einen Designer oder auf eine Verwendung der Entscheidungsroutinen des MA.

Riley und Veloso haben in [53] bereits angedeutet, daß sie in der Zukunft die automatische Generierung von Verhaltens-Modellen verfolgen wollen. Erste Ansätze zur automatischen Generierung von Verhaltens-Modellen kann man in den Arbeiten von Raines u.a. [49] und Wendler u.a. [60] erkennen.

Für einfache MAS existiert ein Verfahren für eine automatische Generierung von Verhaltens-Modellen [7, 8]. Das Verfahren für die dort betrachteten einfachen 2-Agenten Matrix-Spiele läßt sich aber nicht auf komplexe MAS verallgemeinern. Carmel und Markovitch betrachten wiederholte 2-Agenten Matrixspiele in welchen beide Agenten der IP und der MA über das gleiche globale Wissen verfügen. Unter der Annahme, daß der IP als diskreter finiter Automat (DFA) modelliert werden kann, wird basierend auf der Interaktionserfahrung mit dem IP dessen Aktionsauswahl als DFA modelliert. Mit diesem Automat kann dann der MA die folgenden Aktionen des IP vorhersagen. Basierend auf dem Modell des IP bestimmt der MA eine optimale Interaktionsstrategie um seinen Gewinn bei Interaktionen mit dem IP zu maximieren. Die Interaktionsstrategie ist ebenfalls als DFA repräsentiert. Stimmt die Vorhersage der Aktion des IP nicht mit der Realität überein, werden der Modell-Automat des IP und die Interaktionsstrategie entsprechend aktualisiert. Als Ursache des Verhaltens wird hier nicht nur die aktuelle Situation des IP genutzt (gegeben durch den Ausgang der letzten Interaktion) sondern auch die gesamte vorherige Interaktionserfahrung zwischen IP und MA. Letztere ist durch den ermittelten internen Zustand des Modell-DFA repräsentiert.

Konzept zur Generierung eines Verhaltens-Modells: Für die automatische Generierung von Verhaltens-Modellen sind wiederholte Beobachtungen der IP durch den MA nötig. In einem ersten Schritt müssen Aktionen oder Auswirkungen von Aktionen der IP erkannt und zu Verhalten kombiniert werden. Für situationsbasierte Verhaltens-Modelle ist in einem zweiten Schritt zu ermitteln in welchen Situationen konkrete Verhalten durch den IP ausgeführt werden. Paare von Situationen und ausgeführten Verhalten können als Grundlage für ein automatisch generiertes Verhaltens-Modell dienen. Dafür können z.B. verschiedene Lernverfahren genutzt oder auch ein fallbasierter Ansatz (wie in dieser Arbeit) verfolgt werden. Für die Anpassung an mögliche Verhaltensänderungen der IP wird bei jeder Interaktion das Verhaltens-Modell aktualisiert.

Für eine automatische Generierung von situationsbasierten Verhaltens-Modellen trifft man die Annahmen, daß

- die aktuellen Situationen in denen sich die IP befinden durch den MA bestimmbar sind und
- sich die IP in ähnlichen Situationen auch ähnlich verhalten.

2.4 Einordnung dieser Arbeit

Die vorliegende Arbeit beschäftigt sich mit dem automatischen Modellieren von Agenten-Verhalten und liefert Beiträge zur Verhaltens-Bestimmung, zur automatischen Generierung von Verhaltens-Modellen und zur Anwendung von Verhaltens-Modellen um Verhalten vorherzusagen.

War in den Arbeiten aus Abschnitt 2.3.1 (Erkennen und Interpretieren von Verhalten) allein die Verhaltens-Bestimmung automatisiert und wurde in den in Abschnitt 2.3.3 (Anwendungsmöglichkeiten von Verhaltens-Modellen) analysierten Arbeiten zwar die Verhaltens-Bestimmung, die Modell-Generierung und die Modell-Anwendung betrachtet, aber die Modell-Generierung nur manuell vorgenommen, so werden in der vorliegenden Arbeit alle Prozesse automatisiert (siehe auch Tabelle 2.2). Der Hauptbeitrag dieser Arbeit liegt dabei in der automatischen Generierung von Verhaltens-Modellen. Die dafür gemachten Annahmen werden nach der Vorstellung des Modellierungsansatzes in Abschnitt 3.7 auf Seite 40 dargestellt.

	Arbeiten aus Abschnitt 2.3.1	Arbeiten aus Abschnitt 2.3.3	vorliegende Arbeit
Verhaltens-Bestimmung	automatisch	automatisch	automatisch
Modell-Generierung	—	**manuell**	**automatisch**
Modell-Anwendung	—	automatisch	automatisch

Tabelle 2.2: Einordnung verschiedener Verhaltens-Modellierungs-Arbeiten für komplexe MAS

Als Anwendungsszenario wird das virtuelle Fußballspiel verwendet, speziell das durch den *SoccerServer* [46] und Spielerprogramme bereitgestellte komplexe MAS (Details in Abschnitt 3.2 auf Seite 31).

Basierend auf der Klassifizierung der themenverwandten Arbeiten aus Abschnitt 2.3.3.6 auf Seite 21 ergeben sich für diese Arbeit die folgenden Merkmale der Verhaltens-Modellierung: Das Verhaltens-Modell wird **automatisch** erstellt und ist **adaptiv**. Es werden **Agenten**- und **Gruppen**-Verhalten in einem **situationsbasierten** Verhaltens-Modell zur Verhaltens-**Vorhersage** modelliert. Im folgenden werden die Beiträge dieser Arbeit genauer dargelegt.

2.4.1 Verhaltens-Bestimmung

In dieser Arbeit wird ein allgemeines Verfahren zur Erkennung und Interpretation von Verhalten in komplexen MAS vorgestellt. Mit diesem Verfahren können nach einer manuellen Spezifikation unterschiedlichster Agenten- und Multi-Agenten-Verhalten durch **Verhaltens-Muster** die ausgeführten Verhalten der Agenten automatisch erkannt werden.

Die für die Spezifikation von Verhalten verwendete Repräsentation hat Ähnlichkeiten mit den Arbeiten von Voelz u.a. [59] sowie Miene und Visser [41], die

auch Verhalten als aus Ereignissen zusammengesetzt spezifizieren. Im Gegensatz zu diesen Arbeiten definieren wir Verhalten nicht nur durch sie umschließende Ereignisse, sondern definieren auch für die Zeiten zwischen Verhaltens-Start und Verhaltens-Ende Ereignisse. Bei diesem Ansatz besteht ein **Verhalten** immer aus einer **Sequenz von Ereignissen**, die sich gegenseitig ablösen und aus weiteren Ereignissen, die sich mit dieser Sequenz teilweise oder komplett überlappen. Unter der Voraussetzung, daß sich jedes Verhalten mindestens durch eine Ereignis-Sequenz spezifizieren läßt, präsentieren wir einen **allgemeinen Algorithmus** zur Erkennung von Verhalten, das auf oben beschriebene Art und Weise spezifiziert wurde. Die verwendeten Ereignisse können auch eine Dauer haben.

Im Gegensatz zu allen anderen Arbeiten, die sich mit der Verhaltens-Bestimmung beschäftigen, ermitteln wir bei der Erkennung von Verhalten auch die **Eigenschaften von Verhalten** und führen anhand der Eigenschaften **Untergruppen von Verhalten** ein, die über eine Unterscheidung in 'erfolgreich' und 'erfolglos' hinaus gehen. Der Ansatz in dieser Arbeit ermittelt also nicht nur, um was für ein Verhalten es sich handelt, sondern auch **wie** es genau von den IP ausgeführt wird. Desweiteren präsentieren wir **Ergebnisse zur Vollständigkeit und Korrektheit** der Verhaltens-Bestimmung.

2.4.2 Generierung von Verhaltens-Modellen

Durch wiederholte Beobachtungen der IP wird basierend auf deren ausgeführten Verhalten in den verschiedensten Situationen ein teamspezifisches **Verhaltens-Modell** der IP **automatisch generiert und aktualisiert**. Dem Autor sind keine Arbeiten bekannt, in denen etwas Ähnliches für komplexe MAS verfolgt wird.

Das Verhaltens-Modell ist aus Paaren von erkannten Verhalten und deren Auslösern zusammengesetzt. Ein Verhaltens-Auslöser beschreibt die wichtigsten Merkmale der Situation in der das Verhalten von den IP ausgeführt bzw. gestartet wird. Mit der Definition eines Ähnlichkeitsmaßes zwischen Auslösern können auch Verhalten für Situationen vorhergesagt werden, in denen die IP bisher nicht beobachtet wurden. Teile des Ähnlichkeitsmaßes werden dabei automatisch an die IP angepaßt. Für das Verhaltens-Modell wird ein **fallbasierter Ansatz** verfolgt.

2.4.3 Anwendung von Verhaltens-Modellen

Wir benutzen ein **situationsbasiertes Verhaltens-Modell** zur **Vorhersage von Verhalten** der IP in komplexen MAS. Damit können im Gegensatz zu der am nächsten verwandten Arbeit (von Riley und Veloso [53]) basierend auf Situationen konkrete Angaben über das Verhalten der IP gemacht werden.

Dabei werden nicht nur Angaben dazu gemacht, was für ein Verhalten die IP ausführen werden, sondern auch **wie** sie es ausführen werden. Letzteres wird durch die Angabe der Verhaltens-Eigenschaften beschrieben. Es werden Ergebnisse zur **Genauigkeit** der Verhaltens-Vorhersage unter verschiedenen Bedingungen angegeben.

Kapitel 3

Der Modellierungsansatz

Wie im letzten Kapitel herausgearbeitet wurde, erfolgt die Verhaltens-Modellierung durch einen Modellierenden Agenten (MA). Dieser beobachtet die Aktionen seiner Interaktionspartner (IP) und/oder deren Auswirkungen in einer Umgebung (engl. environment). Die IP und der MA interagieren mit der Umgebung. Diese Beziehung ist in Abbildung 3.1 dargestellt. Desweiteren zeigt die Abbildung die beiden Hauptaufgaben des MA:

- die Generierung eines Verhaltens-Modells (engl. generating behavior model) und
- die Anwendung eines Verhaltens-Modells (engl. applying behavior model).

Die Generierung eines Verhaltens-Modells beeinhaltet dabei die Verhaltens-Bestimmung, also das Erkennen und Interpretieren von Verhalten.

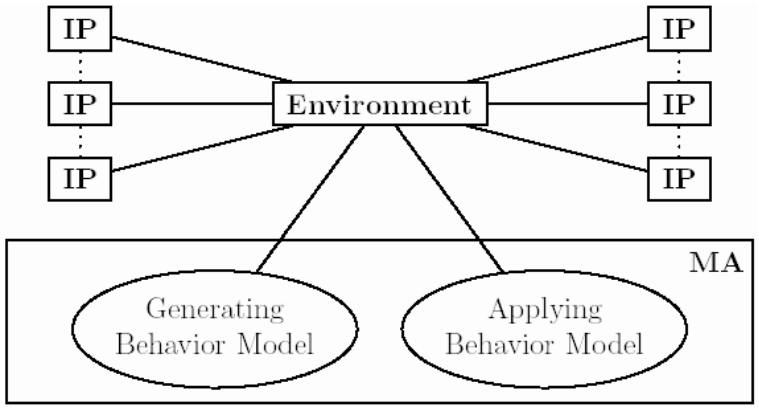


Abbildung 3.1: Struktur des Modellierungsansatzes

In diesem Kapitel wird der in dieser Arbeit verfolgte Ansatz für die automatische Verhaltens-Modellierung beschrieben. Es wird geschildert, aus welchen Teilprozessen sich die Prozesse Verhaltens-Bestimmung, Generierung von Verhaltens-Modellen und Anwendung von Verhaltens-Modellen zusammensetzen.

Wir beginnen mit einer Analyse der Grundlagen für die Verhaltens-Modellierung (Abschnitt 3.1). Danach wird mit dem virtuellen Fußballspiel des

SoccerServer die verwendete Multi-Agenten Umgebung vorgestellt und der Modellierungsansatz für diese Umgebung konkretisiert (Abschnitt 3.2). Die für die Verhaltens-Modellierung verwendeten Begriffe werden eingeführt und definiert (Abschnitt 3.3). Im darauf folgenden Abschnitt werden die Bestandteile der verwendeten Verhaltens-Modelle konkretisiert (Abschnitt 3.4). Anschließend wird der Prozess der Generierung von Verhaltens-Modellen beschrieben (Abschnitt 3.5). Danach widmen wir uns der Anwendung von Verhaltens-Modellen (Abschnitt 3.6), bevor schließlich die Annahmen des Ansatzes zusammengefasst werden. (Abschnitt 3.7).

3.1 Die Grundlagen für die Verhaltens-Modellierung

In diesem Abschnitt werden die Daten vorgestellt, auf deren Grundlage die Modellierung von Verhalten durchgeführt wird. Nach der Einschränkung auf diskrete Zeit (Abschnitt 3.1.1), werden die Begriffe Welt-Situation, Beobachtung und Situation definiert und deren Beziehungen zueinander dargestellt (Abschnitt 3.1.2). Desweiteren wird eine allgemeine Struktur für die IP angegeben (Abschnitt 3.1.3), um später die Annahmen des Modellierungsansatzes besser erläutern zu können.

3.1.1 Diskrete Zeit

Der Ansatz zur Verhaltens-Modellierung basiert auf wiederholten Beobachtungen der IP durch den MA. Eine Beobachtung ist eine Momentaufnahme der Umgebung zu einer bestimmten Zeit, beschreibt also den Zustand der Umgebung. Damit ist die **Zeit** aus Sicht des MA **diskret**. Auch die IP handeln basierend auf Beobachtungen, verwenden also bei der Wahrnehmung auch eine diskrete Zeit. Die Zeiten bei der Aktionsausführung lassen sich ebenso diskretisieren. Wir betrachten daher im folgenden nur Systeme mit diskreter Zeit und sprechen von **Zeitpunkten** oder auch **Zeittakten**.

Die Diskretisierung der Zeit sollte so gewählt sein, daß die Realzeit, die zwischen zwei Zeittakten vergeht, konstant ist. Damit kann von der Dauer zwischen zwei Zeittakten abstrahiert werden. Für den MA ist es weiterhin wichtig, daß der zeitliche Abstand zweier aufeinanderfolgender Zeittakte gering genug ist, damit dieser alle wichtigen Ereignisse wahrnehmen kann. Auch für die IP muß dieser Abstand gering genug sein, damit diese auf Veränderungen der Umwelt schnell genug reagieren können. Der Zeittakt kann dabei künstlich durch das MAS, oder durch die Frequenz der Beobachtungen bestimmt sein.

3.1.2 Welt-Situation, Beobachtung und Situation

Die Gesamtheit der Daten des MAS bildet die Grundlage für Beobachtungen durch die Agenten. Der Gesamtzustand des MAS zu einem bestimmten Zeitpunkt wird mit dem Begriff Welt-Situation bezeichnet.

Definition 3.1 (Welt-Situation) *Eine Welt-Situation W_t beschreibt den Zustand des MAS zu einem konkreten Zeitpunkt t. Dieser Zustand umfasst dabei*

alle beobachtbaren und nicht beobachtbaren Daten der in der Multi-Agenten Umgebung befindlichen Agenten und Objekte.

Zur Welt-Situation gehören damit auch Interna der Agenten, wie z.B. die Energiereserven von Roboter-Agenten. Alle Daten einer Welt-Situation sind per Definition korrekt und vollständig und stellen eine Abstraktion aller im MAS vorhandenen Daten dar. Die Menge aller Welt-Situationen W_t wird mit `worldSituations` bezeichnet.

Die Agenten des MAS nehmen mit Hilfe ihrer Sensoren lediglich Auszüge der Welt-Situationen wahr. Die Daten dieser Auszüge sind zudem fehlerbehaftet. Damit ist es einem Agenten unmöglich eine Welt-Situation exakt nachzubilden, er kann sie nur deuten. Eine Interpretation[1] der Welt-Situation durch einen Agenten wird Beobachtung genannt.

Definition 3.2 (Beobachtung) *Eine Beobachtung O_t^i ist ein interpretierter Ausschnitt einer Welt-Situation W_t durch einen Agenten i. Die Daten einer Beobachtung sind dabei i.a. fehlerbehaftet und unvollständig.*

Der Ausschnitt und die Interpretation sind von der Position des Agenten und von der Art und der Ausrichtung der Sensoren des Agenten abhängig. Damit nehmen zwei beliebige Agenten ihre Umgebung verschieden wahr, erhalten also im allgemeinen aus einer Welt-Situation verschiedene Beobachtungen.

Die Menge aller möglichen Beobachtungen O_t^i wird mit `observations` bezeichnet. Die Interpretation des sichtbaren Ausschnittes der Welt-Situation durch den Agenten i wird mit der Funktion observe dargestellt.

$$\begin{aligned} \text{observe} &: \texttt{worldSituations} \rightarrow \texttt{observations} \\ O_t^i &:= \text{observe}(W_t) \quad \text{durch Agenten } i \end{aligned} \tag{3.1}$$

Abbildung 3.2 auf der nächsten Seite zeigt die Bestandteile einer Beobachtung. Eine Beobachtung enthält den Zeitpunkt zu dem sie generiert wurde, eine Menge von 0 bis n beobachteten Agenten, eine Menge von 0 bis p beobachteten Objekten und eine Menge von 0 bis q Umgebungszuständen[2]. Die beobachteten Agenten und Objekte besitzen dabei mindestens eine Eigenschaft[3], wie z.B. die Position in der Multi-Agenten Umgebung. Jede Eigenschaft enthält dabei neben dem Wert (z.B. der Positionsangabe) auch eine Angabe über die Verläßlichkeit bzw. Genauigkeit des Wertes.

Da die Agenten einen eingeschränkten Sichtbereich haben können, ist es für sie sinnvoll, nicht nur ihre jeweils aktuelle Beobachtung als Grundlage für Handlungsentscheidungen zu nutzen, sondern auch Informationen aus früheren Beobachtungen. Die jeweiligen Daten aller bisherigen Beobachtungen werden deshalb zu einer Situation zusammengefasst.

[1] Unter Interpretation ist die Auswertung aller Sensoren und die Aufbereitung dieser Daten zu einem Zeittakt zu verstehen.

[2] Umgebungszustände sind beobachtbare Daten, die keinem Agenten oder Objekt zugeordnet sind.

[3] Ein Agent bzw. Objekt ohne Eigenschaft könnte gar nicht beobachtet worden sein und kann damit nicht als Bestandteil einer Beobachtung auftreten.

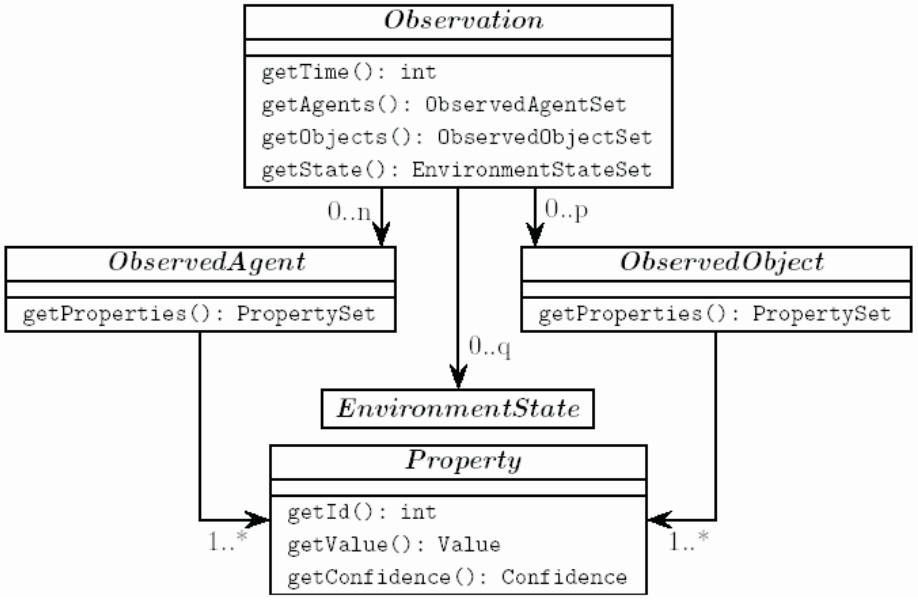


Abbildung 3.2: Bestandteile einer Beobachtung

Definition 3.3 (Situation) *Eine Situation S_t^i ist ein interpretierter Auszug einer Welt-Situation W_t eines Agenten i, welche durch eine Kombination mehrerer Beobachtungen generiert wird.*

Die Beobachtungen stammen dabei von früheren Zeitpunkten oder dem Zeitpunkt t und/oder von anderen Agenten (übermittelt durch Kommunikation). Daten früherer Beobachtungen veralten dabei sukzessive, was durch eine Verringerung von deren Verläßlichkeit ausgedrückt wird.

Demnach ist eine Situation S_t^i eines Agenten i das, was der Agent von der Welt-Situation W_t weiß. Die Menge aller Situationen S_t^i wird mit `situations` bezeichnet. Die Integration der aktuellen Beobachtung O_t^i in die bisher gültige Situation $S_{t^*}^i$ wird mit der Funktion integrate vorgenommen.

$$\begin{gathered} \text{integrate} : \texttt{observations} \times \texttt{situations} \ \rightarrow \texttt{situations} \\ S_t^i := \text{integrate}(O_t^i, S_{t^*}^i) \end{gathered} \tag{3.2}$$

Hierbei gilt, daß die Beobachtung O_t^i aktueller als die Situation $S_{t^*}^i$ ist, d.h. es gilt: $t > t^*$. Prinzipiell ist aber auch eine Integration einer genauso alten oder einer älteren Beobachtung in die aktuelle Situation möglich, z.B. durch (verzögerte) Kommunikation von Situationen durch andere Agenten.

Eine Situation kann damit als eine spezielle, verdichtete Beobachtung angesehen werden. Gegenüber dieser hat eine Situation die zusätzliche Eigenschaft, eine Beobachtung integrieren zu können (siehe auch Abbildung 3.3 auf der nächsten Seite). Für Agenten, die nur die aktuelle Beobachtung als Entscheidungsgrundlage nutzen, gilt: $\text{integrate}(O_t^i, S_{t^*}^i) = O_t^i$.

Auf der Grundlage von wiederholten Beobachtungen und den daraus ermittelten Situationen führt der MA die Verhaltens-Modellierung durch. Wenn die Beobachtungen aus Sicht des MA jeweils vollständig sind (dies ist der Fall, wenn

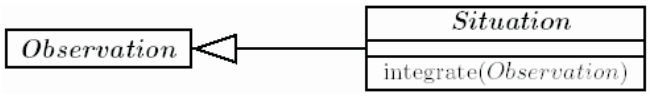


Abbildung 3.3: Situation als spezialisierte Beobachtung

eine neuere Beobachtung die Daten einer früheren Beobachtung vollständig ersetzt), kann der MA seine Beobachtungen auch direkt als Grundlage zur Modellierung von Verhalten benutzen.

3.1.3 Struktur der Interaktionspartner

Ein Teil der Struktur der IP wurde bereits mit den im letztem Abschnitt eingeführten Funktionen angegeben. Mit den Funktionen zur Wahrnehmung (observe) und der Integration von Umgebungsinformationen (integrate) ermittelt jeder IP seine jeweils aktuelle Situation. Basierend auf dieser Situation bestimmt der IP seine nächste Aktion.

Die Menge der Aktionen aller IP wird mit `actions` bezeichnet. Die Ermittlung und Ausführung einer Aktion durch den Interaktionspartner i wird durch die Funktion thinkAndAct repräsentiert.

$$\begin{aligned} \text{thinkAndAct} &: \texttt{situations} \rightarrow \texttt{actions} \\ A_t^i &:= \text{thinkAndAct}(S_t^i) \end{aligned} \tag{3.3}$$

Die Zusammenfassung der Prozesse „think“ und „act“ wurde vorgenommen, um sowohl reaktive Agenten (nur „act“), als auch deliberative Agenten (verwenden beide Prozesse) abzubilden. Aus Sicht des MA ist eine Aufteilung von thinkAndAct nicht nötig. Eine detaillierte Betrachtung der Prozesse von Agenten wird von Burkhard in [5] vorgenommen.

Für die Ermittlung der Aktion kann der IP auch weitere Informationen nutzen, die über die Beschreibung der Situation hinausgehen. Zu solchen Informationen gehören z.B. Verpflichtungen des IP, die dieser u.a. aufgrund früherer Situationen getroffen hat.

Die Aktionen aller Agenten eines MAS führen zu einer neuen Welt-Situation. Dies wird durch die Funktion leadsTo dargestellt.

$$\begin{aligned} \text{leadsTo} &: \texttt{worldSituations} \times \texttt{actions}^k \rightarrow \texttt{worldSituations} \\ W_{t+1} &:= \text{leadsTo}(W_t, A_t^1, \ldots, A_t^k) \end{aligned} \tag{3.4}$$

Dabei bezeichnet k die Anzahl aller Agenten i des MAS. Diese Funktion ist nicht Bestandteil der Struktur der IP sondern Bestandteil der Multi-Agenten Umgebung. Die neue Welt-Situation W_{t+1} ist der Ausgangspunkt für weitere Beobachtungen durch die Agenten.

Die vorgestellte Struktur für die IP ist allgemein gehalten. Abbildung 3.4 auf der nächsten Seite zeigt, wie die Agenten Einfluß auf die Veränderung der Welt-Situation nehmen. Sie nehmen einen interpretierten Auszug der Welt-Situation als Beobachtung wahr, integrieren diese in ihre Situation und reagieren mit der Ausführung von Aktionen. Die Aktionen aller Agenten haben Auswirkungen auf die Veränderung der Welt-Situation.

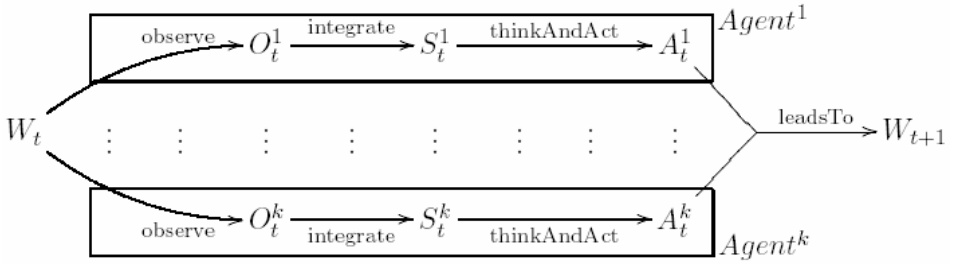


Abbildung 3.4: Einfluß der Agenten auf die Veränderungen der Welt-Situation

Der Prozess integrate ist nicht zwingend notwendig, der IP kann sich auch basierend auf seiner aktuellen Beobachtung für eine Aktion entscheiden (als „stimulus response“ bezeichnet). Damit ist jeder IP mindestens durch die zwei Prozesse observe und thinkAndAct gekennzeichnet. Die Struktur der IP ist für diesen Ansatz weitgehend unbedeutend solange die IP in ähnlichen Welt-Situationen ähnliche Aktionen ausführen. Mit dem Prozess leadsTo werden die Auswirkungen der Aktionen aller Agenten des MAS abgebildet.

3.2 Die verwendete Multi-Agenten Umgebung

Fußball als Anwendungszenario: Wie schon erwähnt, wird als Anwendungsszenario für die Verhaltens-Modellierung das virtuelle Fußballspiel verwendet. Beim Fußball treffen zwei Teams mit entgegengesetzten Zielen aufeinander. Die Spieler beider Teams stellen die IP dar. Als MA wird der Coach verwendet, der das Verhalten aller 22 Spieler auf dem Fußballfeld beobachtet. Der Coach modelliert aber nur eines der beiden Teams (meist ein gegnerisches), d.h. 11 Spieler-Agenten werden modelliert. Diese Beziehungen sind in Abbildung 3.5 dargestellt.

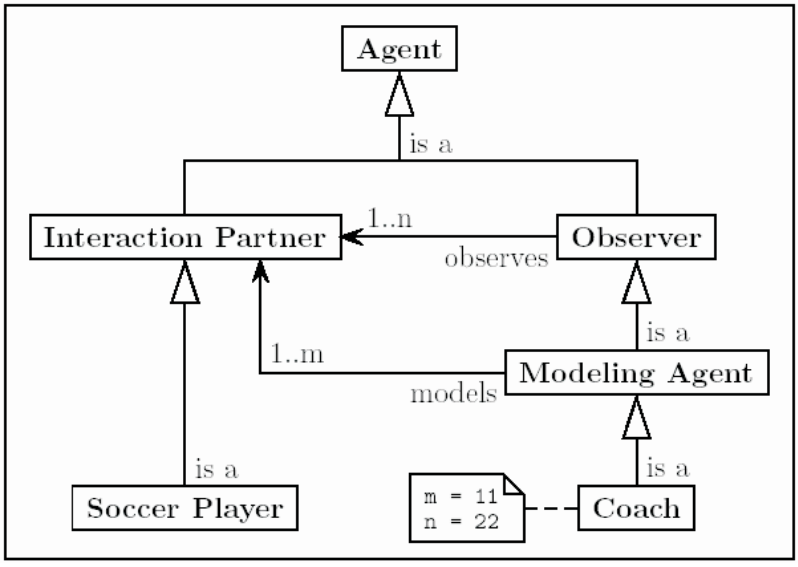


Abbildung 3.5: Fußballspieler und Coach als spezielle Agenten

Damit läßt sich der in Abbildung 3.1 auf Seite 25 dargestellte, allgemeine Modellierungsansatz für die virtuelle Fußballumgebung konkretisieren. Abbildung 3.6 zeigt den konkretisierten Modellierungsansatz. Die Fußballspieler interagieren mit der virtuellen Fußballumgebung und die Verhaltens-Modellierung wird vom Coach für eines der beiden Teams durchgeführt. Welches der beiden Teams modelliert wird, ist dadurch gegeben, ob ein kooperativer oder konkurrenzbasierter Modellierungsansatz verfolgt wird.

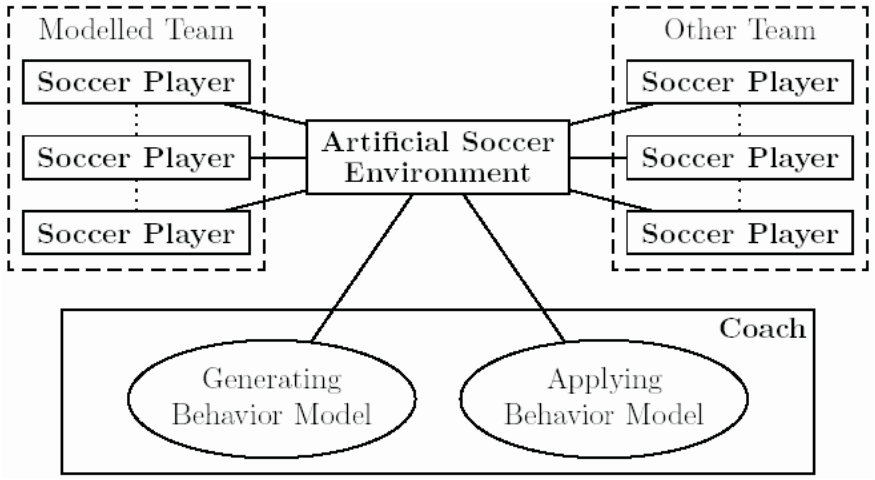


Abbildung 3.6: Struktur des Modellierungsansatzes für die Fußball-Umgebung

Der *SoccerServer*: Die *RoboCup Federation* stellt mit dem *SoccerServer* [46] eine standardisierte Simulationsumgebung für das virtuelle Fußballspiel bereit. Mit dem *RoboCup* [28, 30] wird das Ziel verfolgt, verschiedene Techniken der Künstlichen Intelligenz und der intelligenten Robotik durch die Bereitstellung eines standardisierten Problems zu erproben und zu evaluieren. Dazu werden mindestens jährlich stattfindende Workshops sowie Weltmeisterschaften, Europameisterschaften und andere Turniere durchgeführt.

Mit dem *SoccerServer* wird ein Großteil der Probleme der realen Welt abgebildet. Die Welt-Situationen wechseln beim Fußball extrem schnell. Die Spieler haben häufig nur unscharfe und unvollständige Informationen über ihre Umgebung. Damit hat jeder Spieler unterschiedliche Ansichten über seine Umgebung. Die Spieler müssen schnell handeln und auch schnell auf unerwartete Veränderungen reagieren können. Für ein effizientes Spiel müssen die Spieler einer Mannschaft zusammenarbeiten und dabei die (möglichen) Handlungen der gegnerischen Mannschaft berücksichtigen.

Grundlage für diese Arbeit ist der *SoccerServer* in den Versionen 7.xx der detailliert im *SoccerServer*-Manual [15] beschrieben ist. Die für diese Arbeit relevanten Eigenschaften des *SoccerServer* werden im folgenden dargelegt.

Allgemeine Informationen über den *SoccerServer*: Der *SoccerServer* bildet zusammen mit den Spieler-Agenten ein Client-Server-System. Der *SoccerServer* sendet den Spielern in festgelegten Intervallen Sicht- und Hörinfor-

mationen, worauf die Spieler mit der Übermittlung von Aktionen reagieren. Die Aktionen der Spieler werden vom Server ausgewertet und die Bewegungen des Balles und der Spieler werden auf einem virtuellen Fußballfeld (von der Größe eines richtigen Fußballfeldes) simuliert. Die Simulation folgt vereinfachten physikalischen Gesetzen von Krafteinwirkungen und langsam abklingenden Bewegungen. Es werden dabei auch zufällige Abweichungen eingerechnet. Ein virtueller Schiedsrichter trifft Entscheidungen, wie Einwurf und Abseits.

Durch den *SoccerServer* wird ein diskreter Zeittakt vorgegeben, alle 100 Millisekunden werden die Aktionen der Spieler ausgewertet und eine neue Welt-Situation wird ermittelt. Die wichtigsten Aktionen der Spieler sind beschleunigen, drehen und den Ball kicken, die alle durch den Spieler parameterisiert werden können. Ein Spiel dauert in der Regel 10 Minuten.

Die Beobachtungen der Spieler sind von deren Aktionsausführung entkoppelt. Die Spieler erhalten unter normalen Umständen alle 150 Millisekunden eine neue lokale Sichtinformation und damit eine neue Beobachtung. Dabei nehmen sie alle Spieler und Objekte in einem Ausschnitt von 90 Grad wahr. Die Daten über die Spieler und Objekte sind fehlerbehaftet. Der Fehler ist umso größer, je weiter die Distanz zu den Spielern bzw. Objekten ist.

Der Coach ist ein weiterer Client des *SoccerServer*, der eine globale Sicht auf das Spielfeld hat und der den Spielern seines eigenen Teams Anweisungen und Ratschläge geben kann. Diese Kommunikation unterliegt Einschränkungen hinsichtlich der zu verwendenden Sprache und der Kommunikationshäufigkeit (Frequenz).

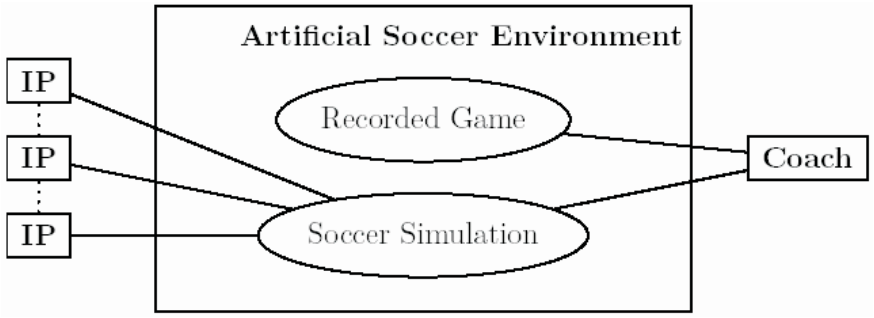


Abbildung 3.7: Anwendungsmöglichkeiten der Fußballumgebung

Die virtuelle Fußballumgebung des *SoccerServer* kann auch zur Wiedergabe von aufgezeichneten Spielen verwendet werden. Diese Beziehung ist in Abbildung 3.7 dargestellt. Der Coach kann damit die Aufzeichnungen früherer Spiele auswerten, anstatt ein gerade stattfindendes Spiel zu beobachten. Mit den Aufzeichnungen aller bedeutenden Turniere stehen genügend Daten für diese Arbeit zur Verfügung.

Spielzustände: Die Arbeitsweise des *SoccerServer* hängt davon ab, in welchem Spielzustand er sich gerade befindet. Nach einem Schuß des Balles ins Aus ist es z.B. nur einer Mannschaft erlaubt, den darauf folgenden Einwurf durchzuführen. Es können grundsätzlich drei Gruppen von Spielzuständen unterschieden werden.

Im Spiel: Dies ist der Spielzustand, in dem das eigentliche Spiel stattfindet, die Spieler beider Mannschaften sich also aktiv um die Kontrolle des Balles bemühen. Im *SoccerServer* wird dieser Spielzustand mit *play on* bezeichnet.

Freistoß etc.: Mit Freistoß etc. werden alle Spielzustände beschrieben, in denen nur ein Team die Möglichkeit hat, den Ball zu kicken, sei es weil der Ball ins Aus geschossen wurde, ein Torwart den Ball gefangen hat, der Ball ins Tor geschossen wurde oder eine Mannschaft einen Freistoß ausführen darf (z.B. als Folge von Abseits).

Aufstellungen: Die Spielzustände dieser Gruppe sind für die Positionierungen der Spieler beider Mannschaften gedacht. Den Mannschaften soll Zeit gegeben werden, damit sie ihre Stellungen einnehmen können. Bei Aufstellungen darf kein Spieler den Ball kicken.

Beobachtungen durch den Coach: Im Gegensatz zu den Spielern hat der Coach eine globale Sicht, er beobachtet das Spiel aus einer Art Vogelperspektive. Der Aufbau einer Beobachtung des Coaches ist in Abbildung 3.8 dargestellt. In der Abbildung sind nur die auch vom Coach für die Verhaltens-Modellierung verwendeten Daten von Spieler und Ball aufgeführt. Eine Beobachtung des Coaches besteht aus 22 Spielern, einem Ball, dem aktuellen Spielzustand und dem nicht in der Abbildung dargestellten Zeittakt. Ein Spieler ist dabei durch seine Position, seine Nummer und sein Team, der Ball durch seine Position und seine Geschwindigkeit beschrieben. Es ist auch möglich auf die Geschwindigkeit des Balles zu verzichten, da diese direkt aus der Positionsdifferenz des Balles ermittelt werden kann. Weitere Daten über die der Coach zwar verfügt, die aber nicht für die Verhaltens-Modellierung verwendet werden, sind Informationen über die Spieler und zwar deren Geschwindigkeit, Körperrichtung und Blickrichtung.

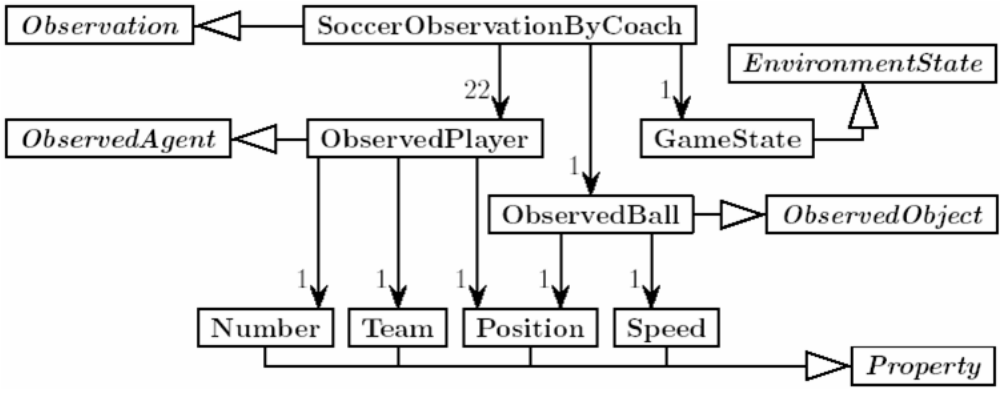


Abbildung 3.8: Bestandteile einer Beobachtung eines Coaches beim Fußball

Im Gegensatz zu den Beobachtungen der Spieler sind die Beobachtungen des Coaches sehr genau. Diese Beobachtungen enthalten mit den 22 Spielern und dem Ball fast alle spielrelevanten Daten über die der *SoccerServer* verfügt.

Lediglich die vom *SoccerServer* verwalteten Daten bzgl. der Kraftreserven der Spieler sind durch den Coach nicht beobachtbar. Neuere Beobachtungen ersetzen die Daten älterer Beobachtungen vollständig. Daher muß der Coach neue Beobachtungen nicht in bestehende Situationen integrieren, sondern kann die Beobachtung direkt als Grundlage für die Verhaltens-Modellierung nutzen. Die Begriffe Beobachtung und Situation werden daher im folgenden synonym verwendet.

3.3 Begriffe für die Verhaltens-Modellierung

Bevor die Prozesse der Generierung und der Anwendung von Verhaltens-Modellen beschrieben werden, führen wir in diesem Abschnitt die dafür benötigten Begriffe ein. Es werden die Begriffe Verhalten, Verhaltens-Muster, Auslöser und Auslöser-Muster definiert. In späteren Kapiteln werden diese Begriffe formal beschrieben.

3.3.1 Verhalten und Verhaltens-Muster

Der Begriff Verhalten wird in der Psychologie als die Gesamtheit der Reaktionen von Individuen auf die Lebensbedingungen ihrer Umwelt [22] bzw. als jede physische Aktivität eines lebenden Organismus, die grundsätzlich von Beobachtern feststellbar ist, verstanden [14]. In der Soziologie steht Verhalten für die Gesamtheit der Lebensäußerungen von Individuen und sozialen Gruppen [22] oder wird als allgemeinste Bezeichnung für jede Aktivität oder Reaktion eines Organismus betrachtet [19].

Solche Verhaltens-Begriffe sind für den hier verwendeten Modellierungsansatz nicht direkt anwendbar. In dieser Arbeit werden stattdessen Verhaltens-Akte, also zeitlich begrenzte Auszüge des Verhaltens von Individuen und (sozialen) Gruppen betrachtet. Diese Verhaltens-Akte werden kurz als Verhalten bezeichnet. Wir beschränken uns dabei auf äußeres, beobachtbares Verhalten von Agenten und Agenten-Gruppen.

Ein Verhalten eines Agenten ist nicht nur durch dessen Aktionen, sondern auch durch die Auswirkungen der Aktionen auf einzelne Objekte und die Umgebung des MAS gekennzeichnet. So gehört zu einem Torschuß-Verhalten nicht nur der Schuß durch den Kicker, sondern auch die Auswirkungen auf den Ball der beschleunigt wird, und schließlich sogar die Auswirkung auf die Umgebung durch Erzielen eines Treffers. Das Torschuß-Verhalten wird durch den Schuß gestartet und endet mit dem Treffer im Tor. Der Kicker ist dabei nur zum Start des Verhaltens am Torschuß-Verhalten beteiligt.

Durch die Aktionen des Agenten und deren Auswirkungen sind Beziehungen zwischen dem Agenten, den beteiligten Objekten und dem MAS vorgegeben. Diese Beziehungen können durch Ereignisse beschrieben werden. Die zwei wichtigsten Ereignisse beim Torschuß sind das Kicken des Balles durch einen Spieler und das darauf folgende Treffen des Tores.

Definition 3.4 (Agent-Verhalten) *Ein Agent-Verhalten ist ein, durch mehrere Ereignisse beschreibbarer, zeitlich begrenzter Ablauf im MAS, der durch die Aktionen eines Agenten und deren Auswirkungen hervorgerufen wird.*

Dadurch, daß zum Agent-Verhalten auch die Auswirkungen von Aktionen des Agenten dazugehören, ist es nicht notwendig, daß der Agent vom Verhaltens-Start bis zum Verhaltens-Ende am Verhalten beteiligt ist. Auch wenn ein Agent nur für einen kurzen Moment mit Aktionen zu einem länger andauernden Verhalten beiträgt, sagen wir, daß der Agent das Verhalten ausführt.

Bei einem Multi-Agenten-Verhalten sind mehrere Agenten an der Ausführung eines Verhaltens beteiligt, d.h. mehr als ein Agent führt Aktionen aus, welche zu Auswirkungen führen, die durch Ereignisse beschrieben werden können. Die Agenten können dabei gleichzeitig, zeitversetzt oder auch überlappend Aktionen ausführen und müssen nicht von Start bis Ende am Verhalten beteiligt sein. Beim Paß-Verhalten z.B. führt erst der Kicker einen Schuß aus, dies hat Auswirkungen auf die Bewegung des Balles, worauf der Empfänger Aktionen ausführt um den Ball abzufangen. Der Kicker ist nach dem Schuß nicht mehr am Paß-Verhalten beteiligt.

Definition 3.5 (Multi-Agenten-Verhalten (abgekürzt Verhalten)) *Ein Verhalten (engl. behavior) ist ein, durch mehrere Ereignisse beschreibbarer, zeitlich begrenzter Ablauf im MAS, der durch die Aktionen mehrerer Agenten und deren Auswirkungen hervorgerufen wird.*

Ein Verhalten kann mit einem Namen und einer Sequenz von Werten zusammengefasst werden. Werte sind dabei die am Verhalten beteiligten Agenten und Objekte (falls nicht schon eindeutig durch den Verhaltensnamen definiert), die Start- und die Endzeit des Verhaltens und wichtige Verhaltens-Eigenschaften. Während bei der Verhaltens-Erkennung (Kapitel 4) ein Verhalten durch Ereignisse beschrieben wird, verwenden wir bei den darauf folgenden Betrachtungen nur die charakteristischen Werte von Verhalten, die in einer Sequenz zusammengefasst werden. Solcherart beschriebene Verhalten werden im folgenden auch **Verhaltens-Essenzen** genannt. Die Menge der Verhaltens-Essenzen wird mit $\texttt{behaviors}_E$ bezeichnet, sie wird später in Formel 5.7 auf Seite 114 definiert.

Damit die Verhalten der IP automatisch erkannt werden können, wird eine Spezifikation der zu erkennenden Verhalten benötigt. Dazu werden Verhaltens-Muster definiert. Ein Verhaltens-Muster fasst die Verhalten gleichen Namens zusammen und abstrahiert von den konkreten Werten eines Verhaltens. Das Verhaltens-Muster wird mit dem Namen der Verhalten, die es repräsentiert, bezeichnet.

Definition 3.6 (Verhaltens-Muster) *Ein Verhaltens-Muster (engl. behavior pattern) ist eine Schablone für eine Menge von Verhalten. Es repräsentiert alle Verhalten gleichen Namens.*

Auf Basis eines Verhaltens-Musters kann der MA Verhalten seiner IP, welche diesem Muster entsprechen, in Beobachtungssequenzen identifizieren und die beteiligten Spieler und Objekte, die Start- und Endzeit sowie wichtige Eigenschaften des Verhaltens bestimmen.

3.3.2 Auslöser und Auslöser-Muster

Bei der in dieser Arbeit verfolgten situationsbasierten Verhaltens-Vorhersage wird davon ausgegangen, daß das ausgeführte Verhalten der IP zu einem Großteil durch deren Situationen zum Startzeitpunkt des Verhaltens bestimmt ist. Diese Situationen nennen wir **Entscheidungs-Situationen**.

Dabei ist nur ein Teil der Daten einer Entscheidungs-Situation für die Auswahl der Aktionen der IP maßgebend. Der für ein Verhalten relevante Teil der Daten einer Entscheidungs-Situation wird Verhaltens-Auslöser oder kurz Auslöser genannt. Beim Torschuß-Verhalten ist z.B. die Position des Schützen und des gegnerischen Torwarts für die Auswahl von Aktionen maßgebend, die Position des Torwarts des eigenen Teams ist aber vollkommen irrelevant.

Definition 3.7 (Auslöser) *Ein Auslöser (engl. trigger) ist der relevante Teil der Daten einer Entscheidungs-Situation für das Zustandekommen eines Verhaltens durch einen oder mehrere Agenten.*

Ein Auslöser kann durch einen Namen und eine Sequenz von Werten zusammengefasst werden. Werte können hier z.B. absolute und relative Spieler- und Ballpositionen sein.

Für die automatische Bestimmung von Auslösern müssen diese vorher spezifiziert werden. Dazu werden alle möglichen Auslöser der Verhalten eines Verhaltens-Musters zu einem Auslöser-Muster zusammengefasst.

Definition 3.8 (Auslöser-Muster) *Ein Auslöser-Muster (engl. trigger pattern) ist eine Schablone für eine Menge von Auslösern. Es repräsentiert die Auslöser aller Verhalten eines Verhaltens-Musters.*

Jedem Verhaltens-Muster ist genau ein Auslöser-Muster zugeordnet. Die Auslöser-Muster dienen als Grundlage für die Bestimmung der Werte von Auslösern in Entscheidungs-Situationen.

Ein Auslöser beschreibt also einen Teil des MAS zu einem Zeitpunkt, während ein Verhalten einen zeitlich begrenzten Ablauf im MAS beschreibt.

3.4 Konkretisierung der Definition von Verhaltens-Modellen

Das mit der Definition 2.4 auf Seite 11 allgemein eingeführte Verhaltens-Modell $\mathfrak{M}$ wird in diesem Abschnitt für den in dieser Arbeit verfolgten Modellierungsansatz konkretisiert.

Es wird ein situationsbasiertes Verhaltens-Modell verwendet, d.h. die Menge $\mathfrak{U}$ ist durch die Menge aller Situationen $\texttt{situations}$ bestimmt. Die Vorhersage von Verhalten wird aber nur für Entscheidungs-Situationen betrachtet, für Situationen die keine Entscheidungs-Situationen darstellen bleibt die Funktion $\mathfrak{f}$ undefiniert. Die Menge $\mathfrak{V}$ ist durch die Menge aller Verhaltens-Essenzen $\texttt{behaviors}_E$ bestimmt. Damit ergibt sich die in Formel 3.5 dargestellte Konkretisierung für Verhaltens-Modelle $\mathfrak{M}$.

$$\mathfrak{M} = [\texttt{situations}, \texttt{behaviors}_E, \mathfrak{f}] \quad \text{mit} \quad \mathfrak{f}\colon \texttt{situations} \rightarrow \texttt{behaviors}_E \tag{3.5}$$

Die Abbildung $\mathfrak{f}$ des Verhaltens-Modells $\mathfrak{M}$ ist in dieser Arbeit durch ein **Fallbasiertes System** gegeben. Dieses System besteht aus einer Fallbasis, gegeben durch eine Menge von Fällen, aus einem Ähnlichkeitsmaß welches zwischen den Fällen definiert ist, aus einem Prozess zur Vorhersage von Verhalten und aus einem Prozess zur Aktualisierung des Verhaltens-Modells. Dabei wird die Technik des Fallbasierten Schließens (FBS) [32, 35] angewendet.

Die Funktionalität der Abbildung $\mathfrak{f}$ wird dabei im wesentlichen durch den Prozess der Verhaltens-Vorhersage realisiert. Die Fallbasis und das Ähnlichkeitsmaß werden bei der Generierung des Verhaltens-Modells erzeugt (siehe Abschnitt 3.5) und von den beiden Prozessen bei der Anwendung des Verhaltens-Modells verwendet (siehe Abschnitt 3.6).

3.5 Generierung eines Verhaltens-Modells

Bevor ein Verhaltens-Modell angewendet werden kann, muß es generiert werden. In diesem Abschnitt wird vorgestellt, wie die automatische Generierung von Verhaltens-Modellen erfolgt.

Als Anlaß für die Ausführung eines Verhaltens durch die IP wird der relevante Teil der entsprechenden Entscheidungs-Situation, also der Auslöser des Verhaltens, betrachtet. Auslöser und Verhalten bilden damit eine Einheit, die als ein Fall verwendet wird. Der Auslöser stellt das Problem und das Verhalten die Lösung eines solchen Falles dar.

$$\boxed{\text{Fall} = (\text{Auslöser, Verhalten})}$$

Die Idee beim FBS ist es, für die Lösung eines neuen Problems nach ähnlichen älteren Problemen zu suchen, die Lösungen der ähnlichsten Probleme an das neue Problem anzupassen und als Lösung zu verwenden. Es wird also ein Ähnlichkeitsmaß benötigt.

Die Generierung eines Verhaltens-Modells durch den MA ist in drei Prozesse gegliedert, die in Abbildung 3.9 auf der nächsten Seite dargestellt sind. Der MA beobachtet seine Umgebung um Auslöser-Verhaltens-Fälle zu erzeugen (engl. generate case). Die erzeugten Fälle werden in eine Fallbasis integriert (engl. integrate case). Damit auch für bisher noch nicht beobachtete Auslöser eine Lösung durch die Fallbasis geliefert werden kann, wird ein Ähnlichkeitsmaß zwischen den Fällen spezifiziert (engl. specify case similarity). Hierbei wird der MA partiell vom Designer unterstützt.

Die beiden wesentlichen Prozesse, die Erzeugung eines Falles (Abschnitt 3.5.1) und die Spezifikation des Ähnlichkeitsmaßes (Abschnitt 3.5.2), werden im folgenden genauer vorgestellt.

3.5.1 Erzeugung eines Falles

Der Prozess der Erzeugung eines Auslöser-Verhaltens-Falles wird weiter strukturiert. In Abbildung 3.10 auf der nächsten Seite ist diese Strukturierung wiedergegeben. Da ein Fall aus einem Auslöser und einem Verhalten besteht,

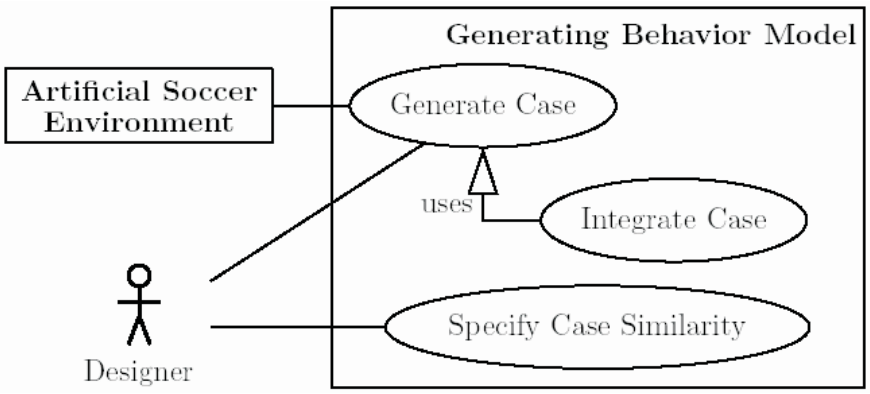


Abbildung 3.9: Struktur der Generierung eines Verhaltens-Modells

müssen zur Erzeugung von Fällen sowohl die Verhalten der IP, als auch die jeweils dazugehörigen Verhaltens-Auslöser erkannt werden (engl. recognize behavior/recognize trigger). Damit die Verhalten und Auslöser automatisch erkannt werden können, müssen diese vorher spezifiziert werden (engl. specify behavior/specify trigger). Dies geschieht durch die Definition von Verhaltens-Mustern und Auslöser-Mustern durch einen Designer. Damit geschieht die Erkennung von Verhalten und Auslösern automatisch, ein Designer muß aber im Vorfeld der Erkennung entsprechende Verhaltens-Muster und Auslöser-Muster spezifizieren.

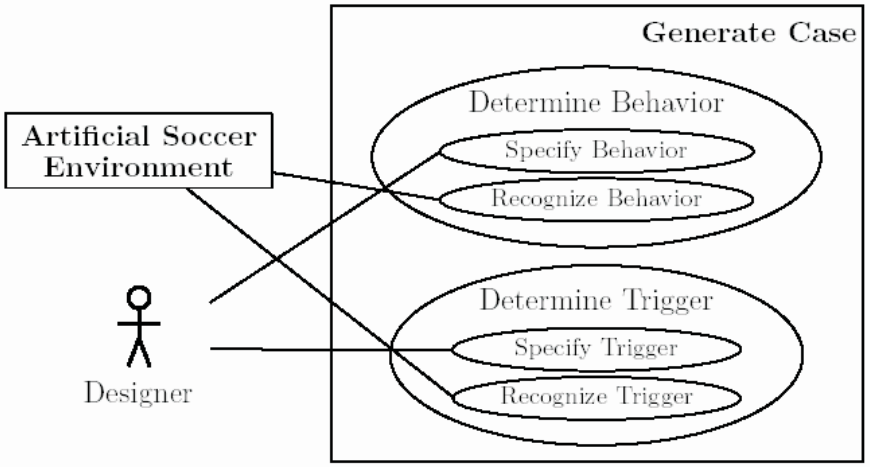


Abbildung 3.10: Struktur der Generierung eines Falles

Die Spezifikation und die Erkennung von Verhalten bilden zusammen den Prozess der **Verhaltens-Bestimmung** (engl. determine behavior). Diesem wichtigen Prozess ist das gesamte Kapitel 4 gewidmet. Die Spezifikation und die Erkennung von Auslösern bilden zusammen den Prozess der **Auslöser-Bestimmung** (engl. determine trigger), der in den Abschnitten 5.1 und 5.2 behandelt wird. Die Kombination von Auslösern und Verhalten zu Fällen wird in Abschnitt 5.4.1 genauer betrachtet.

3.5.2 Spezifikation des Ähnlichkeitsmaßes

Es werden zwei Ähnlichkeitsmaße benötigt, eines, welches die Ähnlichkeit zwischen Auslösern definiert und eines für die Ähnlichkeit zwischen Verhalten. Ein Ähnlichkeitsmaß zwischen Verhalten wird benötigt, um die Vorhersage automatisch evaluieren zu können und um im Bedarfsfall das Verhaltens-Modell um neue Fälle zu erweitern. Bei der Anfrage an die Fallbasis ist aber nur das neue Problem, in Form eines Auslösers, bekannt. Es wird daher für die Vorhersage von Verhalten ein Ähnlichkeitsmaß zwischen Auslösern benötigt.

Da die Auslöser und Verhalten jeweils durch einen Namen und einer Sequenz von Werten zusammengefasst werden können, wird jeweils ein zusammengesetztes Ähnlichkeitsmaß verwendet. Jedes zusammengesetzte Ähnlichkeitsmaß besteht aus lokalen Ähnlichkeiten zwischen den Werten von Auslösern bzw. Verhalten und einer gewichteten Summe (spezielles Relevanzmaß) dieser lokalen Ähnlichkeiten. Beim Ähnlichkeitsmaß der Verhalten werden die lokalen Ähnlichkeiten und die Gewichte durch einen Designer festgelegt. Beim Ähnlichkeitsmaß der Auslöser werden nur die lokalen Ähnlichkeiten durch den Designer festgelegt, die Gewichte werden dagegen automatisch ermittelt.

Konkrete Angaben zur Spezifikation und Ermittlung des Ähnlichkeitsmaßes werden in Abschnitt 5.5 gegeben.

3.6 Anwendung eines Verhaltens-Modells

Die Anwendung eines Verhaltens-Modells gliedert sich in zwei Prozesse, die in Abbildung 3.11 dargestellt sind. Das Verhaltens-Modell wird in Entscheidungs-Situationen zur Vorhersage von Verhalten der IP genutzt (engl. predict behavior). Da es bei der Verhaltens-Vorhersage auch zu Fehlern kommen kann, müssen fehlerhafte Vorhersagen erkannt und das Verhaltens-Modell gegebenenfalls an die Verhaltensänderungen der IP angepaßt werden (engl. revise behavior model).

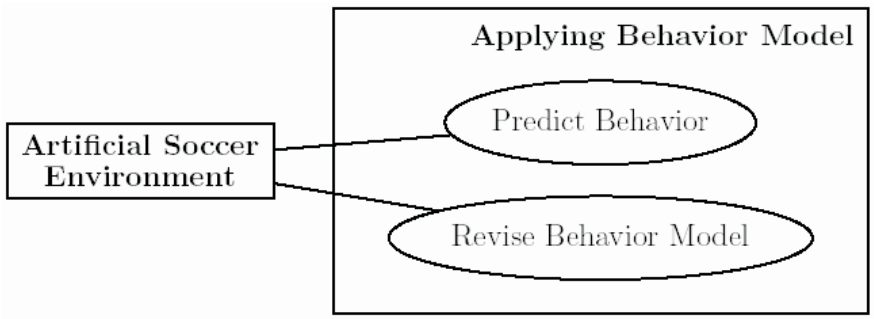


Abbildung 3.11: Struktur der Verwendung eines Verhaltensmodells

Für die Verhaltens-Vorhersage werden bei Auftreten einer Entscheidungs-Situation die geltenden Auslöser bestimmt[4]. Unter Nutzung der Fallbasis und des Ähnlichkeitsmaßes für die Auslöser werden diejenigen Fälle ermittelt, deren

[4]Die Bestimmung mehrerer Auslöser ist notwendig, da die IP verschiedene Verhalten ausführen können.

Auslöser den Auslösern der Entscheidungs-Situation am ähnlichsten sind. Auf Basis dieser Fälle ermittelt der MA das durch die IP wahrscheinlich ausgeführte Verhalten.

Das Verhaltens-Modell wird angepaßt, indem der MA die IP beobachtet, deren tatsächlich ausgeführtes Verhalten bestimmt und es mit dem vorhergesagten Verhalten vergleicht. Wenn die Vorhersage fehlerhaft war, wird dies als Indiz für eine unvollständige Fallbasis angesehen und das gerade wahrgenommene Auslöser-Verhaltens-Paar in die Fallbasis aufgenommen.

Beim Vergleich der Verhaltens-Vorhersage mit den im Datenbanken-Bereich verwendeten Event-Condition-Action-Rules [10, 11] (dt. Ereignis-Bedingungs-Aktions-Regeln) entspricht dem Event das Auftreten einer Entscheidungs-Situation, der Condition ein Auslöser und der Action ein vorhergesagtes Verhalten.

In Kapitel 6 wird die Anwendung des Verhaltens-Modells zur Verhaltens-Vorhersage genau behandelt.

3.7 Annahmen des Ansatzes

Die Grundvoraussetzung der situationsbasierten Verhaltens-Vorhersage ist, daß die Verhalten der IP im wesentlichen durch deren Situationen bestimmt sind. Dabei sollen die IP in ähnlichen Situationen auch ähnliche Verhalten ausführen. Falls diese Annahme nicht erfüllt ist, z.B. weil die IP zufällig handeln, macht dieser Ansatz nur wenig Sinn.

Im folgenden werden die für den Ansatz dieser Arbeit gemachten fünf Annahmen vorgestellt. Nach einer Kurzvorstellung der jeweiligen Annahme wird sie durch Bemerkungen erläutert.

Annahmen:

1. Gegeben ist eine nichtleere, endliche **Menge von Verhaltens-Mustern**.

   Die Repräsentation der Handlungen der IP durch Verhaltens-Muster unterliegt subjektiven Entscheidungen, so gibt es z.B. unterschiedliche Betrachtungsweisen was unter einem Paß im Fußball zu verstehen ist. Die Verhaltens-Muster sollten so gewählt werden, daß deren Verhalten durch den MA beobachtet und als Umsetzung zum Erreichen von Zielen oder Teilzielen der IP angesehen werden können. Es ist dabei nicht nötig, daß die Ziele im IP repräsentiert sind, es reicht wenn die IP durch ihre Handlungen auf diese hinarbeiten. So kann das Verhaltens-Muster Torschuß uneingeschränkt als Umsetzung des Zieles Tore zu schießen angesehen werden.

2. Ursache für ein Verhalten ist die Situation zum Startzeitpunkt des Verhaltens, diese **Entscheidungs-Situation** ist durch den MA **bestimmbar**.

   Die Bestimmung von Entscheidungs-Situationen ist im allgemeinen nicht trivial. So kann ein MA z.B. nicht sicher sagen, wann sich eine Gruppe von IP für die Ausführung einer Abseitsfalle entschieden hat. Wir beschränken

uns daher auf Verhalten mit eindeutig bestimmbarem Startzeitpunkt. So werden nur Verhalten betrachtet, an deren Start ein Spieler die Kontrolle über den Ball hat, wie z.B. beim Paß.

3. Gegeben ist eine nichtleere endliche **Menge von Auslöser-Mustern**, die jeweils einen beobachtbaren Teil der Daten der jeweiligen Entscheidungs-Situationen repräsentieren.

   Die Situationen aller Agenten zu einem Zeitpunkt sind im allgemeinen paarweise verschieden. Die genaue Bestimmung der Situation eines Agenten durch einen MA stellt bei den meisten komplexen MAS ein sehr kompliziertes Problem dar. Dazu gehört nicht nur die schwierige Interpretation der aktuellen und älterer Beobachtungen[5] des IP, sondern auch Wissen über die Arbeitsweise der integrate Funktion (Formel 3.2) des IP und Wissen über kommunizierte Situationsfragmente zwischen den IP.

   Aus diesem Grunde beschränken wir uns auf die Annahme, daß sich die von den IP verwendeten Auslöser in ihren Daten nicht zu stark von dem durch den MA bestimmten Auslöser unterscheiden. Diese Annahme kann durchaus zu Fehlern bei der automatischen Generierung des Verhaltens-Modells und damit zu Fehlern bei der Verhaltens-Vorhersage führen.

   Verschiedene Teams von IP verwenden möglicherweise verschiedene Daten einer Entscheidungs-Situation als Auslöser. Bei der Wahl der Auslöser-Muster für den MA sollte der Designer eher den Auslöser über- als unterspezifizieren. Nicht- oder nur schwach verwendete Auslöser-Daten werden durch diesen Ansatz automatisch erkannt und entsprechend behandelt.

4. Eine **Menge von Auslöser-Verhaltens-Fällen** ist gegeben bzw. bestimmbar.

   Zur Generierung der Verhaltens-Modelle stehen Aufzeichnungen der Spiele von Welt-Meisterschaften, Europa-Meisterschaften und anderer Turniere zur Verfügung. Hier haben zumindest die besseren Teams genügend Spiele gegen unterschiedliche Gegner gespielt. Es ist anzunehmen, daß diese Teams für eine repräsentative Auswahl von Situationen mit der Ausführung von Verhalten reagiert haben. In verschiedenen Szenarien wird untersucht, wieviele Spiele zum Aufbau eines Verhaltens-Modells notwendig sind.

5. Ein **Ähnlichkeitsmaß** für Auslöser und Verhalten ist gegeben, so daß bei ähnlichen, durch den MA beobachtbaren Auslösern, die IP auch ähnliche Verhalten ausführen.

   Mit dieser Annahme wird die Forderung, daß die IP in ähnlichen Situationen auch ähnliche Verhalten ausführen, konkretisiert. Die Genauigkeit der Annahme hängt davon ab wie die Ähnlichkeit zwischen den Auslösern und zwischen den Verhalten definiert ist.

[5] Zur genauen Interpretation von Beobachtungen anderer Agenten benötigt der MA Wissen über den Beobachtungsbereich und die Genauigkeit der Sensoren der IP und Wissen über die aktuelle Ausrichtung der Sensoren der IP.

In dieser Arbeit wird die Ähnlichkeit zwischen den Auslösern teilweise automatisch in Abhängigkeit von der Ähnlichkeit der Verhalten und den gesammelten Auslöser-Verhaltens-Fällen ermittelt, damit diese Annahme besser erfüllt werden kann.

Für die Verhaltens-Bestimmung muß nur die Annahme 1 erfüllt sein, für die Generierung eines Verhaltens-Modell werden die Annahmen 1, 2, und 3 vorrausgesetzt und für eine erfolgreiche Anwendung eines Verhaltens-Modells müssen alle fünf Annahmen erfüllt sein.

Kapitel 4

Die Verhaltens-Bestimmung

Das Ziel des Prozesses der Verhaltens-Bestimmung ist es, das Verhalten von Interaktionspartnern durch Beobachtung der Auswirkungen ihrer Handlungen zu erkennen und zu interpretieren. Es gilt also herauszufinden, was für ein Verhalten ein Agent oder eine Gruppe von Agenten gerade ausführt bzw. ausgeführt hat. Als Grundlage zur Erkennung von Verhalten dienen Verhaltens-Muster, die jeweils eine Schablone für eine Menge von Verhalten darstellen. Die Verhaltens-Bestimmung wird anhand des virtuellen Fußballspiels (vgl. Abschnitt 3.2 auf Seite 30) vorgenommen, so daß als MA der Coach und als IP die Spieler-Agenten verwendet werden.

Nach der Einführung einiger Bezeichnungen (Abschnitt 4.1) wird mit einer ausführlichen Erarbeitung des Verhaltens-Musters Paß begonnen (Abschnitt 4.2). Damit steht ein praktisches Beispiel für die daran anschließende Formalisierung der Verhaltens-Bestimmung (Abschnitt 4.3) zur Verfügung. Danach werden weitere fünf Verhaltens-Muster konzipiert (Abschnitt 4.4). Schließlich wird eine effiziente, objektorientierte Realisierung der Verhaltens-Bestimmung vorgestellt (Abschnitt 4.5). Die Evaluierung der Verhaltens-Bestimmung ist in Kaptitel 7 zu finden.

4.1 Verwendete Bezeichnungen

Für einige der Bestandteile von Coach-Beobachtungen (vgl. Abbildung 3.8 auf Seite 33) werden Mengen-Bezeichnungen eingeführt. So werden Mengen für die Zeitpunkte und die Spieler einer Beobachtung benötigt. Da es beim Fußballspiel mit dem Ball nur ein Objekt und mit dem Spielzustand nur einen Umgebungszustand gibt, werden keine Mengen für Objekte und Umgebungszustände benötigt. Für die Positions- und Geschwindigkeitseigenschaften von Spieler und Ball wird eine Menge von Vektoren eingeführt.

$\mathbb{T}$ ist die Menge der **Zeitpunkte** eines Spieles, $\mathbb{T} \subseteq \mathbb{N}$, wobei ein Spiel im Normalfall (ohne Verlängerung) 6000 Zeittakte lang ist. Mit den Variablen $t_i \in \mathbb{T}$ werden jeweils beliebige Zeitpunkte bezeichnet. Im folgenden wird der Begriff Zeittakt synonym zum Begriff Zeitpunkt benutzt. Für eine Folge von Zeitpunkten t_i mit $t_j \leq t_i \leq t_k$ wird auch der Begriff Zeitintervall verwendet.

$\mathbb{P}$ ist die Menge der **Spieler-Agenten**, $\mathbb{P} = \{\mathrm{mp}_1, \ldots, \mathrm{mp}_{11}, \mathrm{op}_1, \ldots, \mathrm{op}_{11}\}$, wobei die Spieler mp_j (modelled player j) die Agenten des zu modellierenden Teams sind und die Spieler op_k (other player k) die Agenten des anderen, nicht näher betrachteten, Teams sind. Die im folgenden verwendeten Variablen $p_i \in \mathbb{P}$ bezeichnen jeweils einen beliebigen Spieler-Agenten.

$\mathbb{V}$ ist die Menge von zweidimensionalen **Vektoren**, $\mathbb{V} = \mathbb{R} \times \mathbb{R}$. Sie werden für absolute und relative Positionen sowie für Geschwindigkeiten benutzt. Vektoren werden jeweils unterschiedlich, entsprechend ihrer Bedeutung bezeichnet. Für einen Vektor $v \in \mathbb{V}$ bezeichnen $v.x$ und $v.y$ dessen x- und y-Koordinate sowie $v.length$ und $v.angle$ dessen Länge und Richtung. Mit $\measuredangle(v_1, v_2)$ wird der kleinere Winkel zwischen zwei Vektoren $v_1, v_2 \in \mathbb{V}$ bezeichnet.

Im weiteren Verlauf werden die variablen Indizes i, j, k, und l mehrmals zur Unterscheidung von Spieler-Agenten und Zeitpunkten benutzt. Nur im selben Absatz oder derselben Formel stellen gleiche Indizes auch gleiche Spieler bzw. Zeitpunkte dar.

4.2 Das Verhaltens-Muster Paß

Der Transfer einer Ressource von einem Produzenten zu einem Verbraucher ist eines der grundlegenden Koordinationsprobleme in der Koordinationstheorie [38, 39]. Ein Paß ist ein Spezialfall eines solchen Transfers mit Spieler-Agenten als Produzent und Verbraucher und dem Ball als Ressource. Der Kicker[1] des Balles tritt dabei in die Rolle des Produzenten und der Empfänger in die Rolle des Verbrauchers. Damit ist das Passen ein Koordinationsproblem von mindestens zwei Spielern.

In diesem Abschnitt werden die notwendigen Bedingungen für das Paßverhalten herausgearbeitet. Hierzu wird der Transfer eines Balles als Grundlage für einen Paß untersucht (Abschnitt 4.2.1). Danach wird diskutiert, was unter der Kontrolle über den Ball genau zu verstehen ist und damit genau festgelegt, wie ein Balltransfer beginnt und endet (Abschnitt 4.2.2). Aufbauend auf dem Verhaltens-Muster Balltransfer werden weitere notwendige Bedingungen für das Paß-Verhalten erarbeitet und das Verhaltens-Muster Paß angegeben (Abschnitt 4.2.3). Schließlich werden die durch dieses Verhaltens-Muster beschriebenen Pässe weiter in Kategorien unterteilt, indem mögliche Ausprägungen von Pässen ermittelt werden (Abschnitt 4.2.4).

4.2.1 Transfer eines Balles als Basis für einen Paß

Für einen Balltransfer zwischen einem Kicker und einem Empfänger ergeben sich zunächst zwei wesentliche Bedingungen:

- der Ball befindet sich zu einem Zeitpunkt t_0 unter Kontrolle des Kickers p_1, und

[1] Mit Kicker wird in dieser Arbeit ein den Ball schießender Spieler bezeichnet.

- der Ball befindet sich zu einem späteren Zeitpunkt t_n unter Kontrolle des Empfängers p_2.

Damit sichergestellt werden kann, daß p_1 den Ball gekickt und p_2 der alleinige Empfänger des Balles ist, wird den beiden Bedingungen hinzugefügt, daß p_1 und p_2 zum Zeitpunkt t_0 bzw. t_n die **alleinige Kontrolle** über den Ball haben.

Während die Annahme eines Transfers durch den Spieler p_2 augenblicklich erfolgt, kann sich die Vorbereitung eines Transfers durch p_1 auch über mehrere Zeittakte erstrecken. Zur Vorbereitung eines Balltransfers sind dabei 2 Takte ausreichend[2]. Aus diesem Grund wird die jeweilige **Dauer der Ballkontrolle** beider Spieler **beschränkt**, bei Spieler p_1 auf 2 Takte und bei Spieler p_2 auf einen Takt. Damit gehören maximal nur die letzten beiden Takte, in denen der Spieler p_1 die Kontrolle über den Ball hatte, zum Transfer dazu. Ebenso gehört nur der erste Takt, in dem der Spieler p_2 den Ball kontrolliert, zum Transfer.

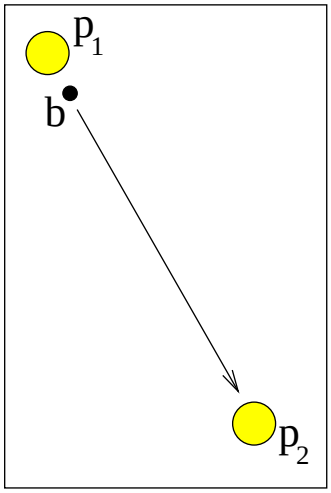


Abbildung 4.1: Transfer des Balles b von Spieler p_1 zu Spieler p_2

Die alleinige Kontrolle eines Spielers p_i über den Ball für einen auf s Takte begrenzten Zeitraum ($s \geq 1, s \in \mathbb{N}$), der im Zeittakt t_j startet und im Zeittakt t_k endet, wird mit dem Ausdruck

$$\text{XBallControl}_\text{s}\,(p_i, t_j, t_k).$$

bezeichnet[3], wobei die Beziehung $t_k - t_j < s$ gilt.

Im allgemeinen können an einem Transfer auch mehr Agenten als p_1 und p_2 beteiligt sein. Wir beschränken uns jedoch auf einen einfachen Transfer. Dies führt zu der Bedingung, daß zwischen der Ballkontrolle durch den Kicker und der Ballkontrolle durch den Empfänger **kein Spieler** den **Ball kontrolliert**. Diese Bedingung wird mit dem Ausdruck

[2]Die 2 Takte zur Vorbereitung eines Transfers orientieren sich an den Gegebenheiten der verwendeten virtuellen Fußballumgebung. Hier können alle Transfers durch eine maximal zweitaktige Manipulation der Ballbewegung erreicht werden, ein Takt ist nicht immer ausreichend.

[3]Das "Xïn $\text{XBallControl}_\text{s}$ steht dabei für das englische ëXclusive".

$$\text{ballFree}\,(t_j, t_k)$$

bezeichnet, wobei t_j und t_k das Zeitintervall der Bedingung begrenzen und $t_j \leq t_k$ gilt.

Auf Grundlage dieser Überlegungen ergeben sich fünf Bedingungen für einen Balltransfer (siehe auch Abbildung 4.1 auf der vorherigen Seite), die im Verhaltens-Muster 1 dargestellt sind. Für eine Zusammenfassung dieses Verhaltens-Musters sind nur der Kicker, der Empfänger, der Start- und der Endzeitpunkt wichtig, von den anderen Zeitpunkten wird abstrahiert.

Verhaltens-Muster 1: Balltransfer von Spieler p_1 zu Spieler p_2

$$\text{ballTransfer}\,(p_1, p_2, t_0, t_n) :- \text{XBallControl}_2\,(p_1, t_0, t_1),\ \text{ballFree}\,(t_2, t_3),\ \text{XBallControl}_1\,(p_2, t_n, t_n),\ \text{follow}\,(t_1, t_2),\ \text{follow}\,(t_3, t_n).$$

Die ersten drei Bedingungen geben die oben beschriebenen Sachverhalte wieder. Die anderen beiden Bedingungen legen die **zeitliche Reihenfolge** der ersten drei Bedingungen fest,

$$\text{follow}\,(t_j, t_k)$$

bedeutet, daß der Zeitpunkt t_k direkt auf den Zeitpunkt t_j folgt, d.h. $t_k = t_j + 1$.

4.2.2 Was bedeutet Ballkontrolle?

In diesem Abschnitt wird erörtert, was Kontrolle eines Spielers über den Ball genau bedeutet. Damit werden die beiden Bedingungen $\text{XBallControl}_\text{s}\,(p_i, t_j, t_k)$ und $\text{ballFree}\,(t_j, t_k)$ genauer erklärt.

Abbildung 4.2 auf der nächsten Seite illustriert das Problem graphisch. Der Spieler p_1 kickt den Ball in Richtung des Spielers p_2. Dabei bewegt sich der Ball durch den Bereich, in dem Spieler p_3 den Ball kicken kann (Kickbereich von p_3 dargestellt durch den gestrichelten Kreis, Ballposition dargestellt durch den nicht gefüllten kleinen Kreis). Der Ball bewegt sich weiter auf Spieler p_2 zu und kommt dort an. Es stellt sich die Frage: Handelt es sich bei diesem Szenario um einen oder um zwei Pässe?

Für einen Beobachter gibt es 2 Varianten, was Kontrolle über den Ball durch einen Spieler bedeuten kann. Diese beiden Varianten werden mit passiver bzw. aktiver Ballkontrolle bezeichnet.

Passive Ballkontrolle: Eine passive Ballkontrolle durch einen Spieler ist gegeben, wenn der Spieler die Möglichkeit hatte die Ballbewegung zu verändern, d.h. der Ball war im Kickbereich des Spielers. Bei dieser Variante handelt es sich bei dem Szenario aus Abbildung 4.2 um zwei Pässe (von p_1 zu p_3 und von p_3 zu p_2), da der Ball sich kurzzeitig im Kickbereich des Spielers p_3 befand.

Aktive Ballkontrolle: Eine aktive Ballkontrolle durch einen Spieler ist gegeben, wenn der Spieler nicht nur die Möglichkeit hatte die Ballbewegung zu verändern, sondern sie auch wahrnehmbar verändert hat. Bei dem in Abbildung 4.2 dargestellten Szenario wurde die Ballbewegung durch Spieler

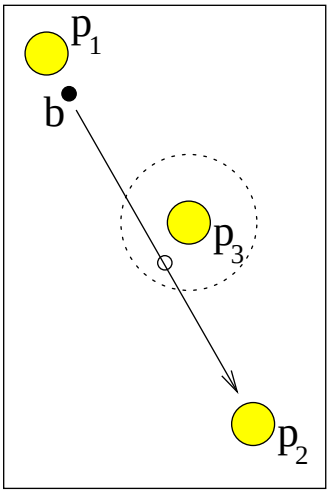


Abbildung 4.2: Ballkontrolle durch Spieler p_3?

p_3 nicht wahrnehmbar verändert (in der Abbildung ist nur die nicht vorhandene Richtungsänderung dargestellt). Damit wurde der von Spieler p_1 begonnene Paß nicht durch Spieler p_3 beendet, sondern erst durch Spieler p_2. Bei dieser Variante handelt es sich also beim dargestellten Szenario um einen Paß (von p_1 zu p_2).

Prinzipiell ist es möglich, sowohl die aktive als auch die passive Ballkontrolle zu verwenden. Dabei umfasst die aktive die passive Ballkontrolle, d.h. wird eine aktive Ballkontrolle erkannt, so gilt auch eine passive Ballkontrolle. Im folgenden wird erarbeitet, welche dieser beiden Varianten bei Beobachtungen durch den Coach besser geeignet ist.

Im Sinne der verwendeten Definition von Verhalten (vgl. Definition 3.5 auf Seite 35) müssen Kicker und Empfänger mit Aktionen zu diesem beitragen. Dies ist bei Verwendung der passiven Ballkontrolle nicht unbedingt erfüllt, aber bei der aktiven Ballkontrolle. Damit ist prinzipiell die Verwendung der aktiven Ballkontrolle zu bevorzugen.

Die Erkennung der passiven Ballkontrolle ist einfacher zu realisieren, da hier nur getestet werden muß, ob ein Spieler existiert, in dessen Kickbereich sich der Ball befindet. Zur Überprüfung ob der Spieler auch die Ballbewegung wahrnehmbar verändert hat, muß der aktuelle Ballgeschwindigkeitsvektor mit dem Ballgeschwindigkeitsvektor des folgenden Taktes verglichen werden. Dazu muß bereits die Beobachtung des folgenden Taktes zur Verfügung stehen.

Die wesentliche Grundlage für die Entscheidung für eine der beiden Varianten ist die Genauigkeit mit der die folgenden Informationen wahrgenommen werden:

(a) ob sich der Ball im Kickbereich eines Spielers befindet und

(b) ob die Ballbewegung durch einen Spieler verändert wurde.

Ist die Wahrnehmungsgenauigkeit von (a) relativ hoch gegenüber der Wahrnehmungsgenauigkeit von (b), so trägt die Verwendung des Konzeptes der aktiven Ballkontrolle nur zu zusätzlichen Erkennungsfehlern gegenüber der passiven

Ballkontrolle bei. Bei der Beobachtung durch den Coach als MA sind jedoch beide Wahrnehmungsgenauigkeiten sehr hoch. So kann relativ genau bestimmt werden, ob die Ballbewegung durch einen Spieler verändert wurde[4] und damit ob ein Spieler tatsächlich mit einer Aktion zum Verhalten beigetragen hat. Damit zeichnet sich bei der Beobachtung durch den Coach[5] die aktive gegenüber der passiven Ballkontrolle aus.

Aufgrund dieser Überlegungen entscheiden wir uns für das Konzept der **aktiven Ballkontrolle**[6]. Basierend auf dieser Entscheidung werden im folgenden die Bedingungen $\text{XBallControl}_\text{s}\,(p_i, t_j, t_k)$ und $\text{ballFree}\,(t_j, t_k)$ genauer erklärt.

Die alleinige Ballkontrolle $\text{XBallControl}_\text{s}\,(p_i, t_j, t_k)$ durch einen Spieler p_i für ein Zeitintervall (t_j, t_k) ist damit gegeben, wenn sich der Ball während dieser Zeit nur im Kickbereich vom Spieler p_i befindet (also nicht auch im Kickbereich von anderen Spielern) und mindestens vom Zeittakt t_j zum Zeittakt $t_j + 1$ eine relevante (über eine zufällige Ballbewegungsänderung hinausgehende) Bewegungsänderung des Balles beobachtet werden konnte.

Die Bedingung $\text{ballFree}\,(t_j, t_k)$ (kein Spieler kontrolliert den Ball im Zeitintervall (t_j, t_k)) besagt, daß es keinen Zeittakt t mit $t_j \leq t \leq t_k$ gibt, in dem sich der Ball im Kickbereich mindestens eines Spielers befindet und für den eine relevante Bewegungsänderung des Balles gegenüber Zeittakt $t + 1$ beobachtet werden konnte. Um Überlappungen mit der Bedingung $\text{XBallControl}_\text{s}\,(p_i, t_j, t_k)$ zu vermeiden, muß weiterhin gelten, daß sich der Ball im Zeittakt t_j im Kickbereich keines Spielers befindet.

Die Größe der Kickbereiche der Spieler und die maximal zufällige Ballbewegungsänderung sind durch den *SoccerServer* vorgegeben.

4.2.3 Vom Balltransfer zum Paß

Der in Abschnitt 4.2.1 eingeführte Balltransfer von einem Kicker zu einem Empfänger dient als Grundlage für die Spezifikation von Paß-Verhalten. Es liegt sogar nahe, den eingeführten Balltransfer bereits als Paß zu bezeichnen.

Für einen Paß sollen sich jedoch beide Spieler aktiv am Transfer beteiligen. Dies bedeutet, daß das in Abbildung 4.3 auf der nächsten Seite dargestellte Szenario nicht als Paß angesehen wird. Hier entfernt sich der Spieler p_1 vom Ball b, der Spieler p_2 bewegt sich auf den Ball zu, und der Ball selber bewegt sich nicht oder nur wenig. Bei diesem Szenario hat p_1 nicht aktiv am Transfer teilgenommen.

Um solche Szenarien für Pässe auszuschließen, wird eine neue Bedingung

[4] Die Genauigkeiten für Positionen und Geschwindigkeiten von Ball und Spieler sind nahezu exakt. Damit kann auch die Geschwindigkeitsänderung des Balles sehr genau ermittelt werden. Diese wird aber auch durch zufällige Abweichungen beeinflusst. Eine Bewegungsänderung des Balles durch einen Spieler innerhalb der Variationsbreite der zufälligen Abweichungen kann damit nicht erkannt werden. Diese Variationsbreite ist jedoch relativ gering.

[5] Bei Beobachtung durch einen Spieler-Agenten ist eventuell die Verwendung der passiven Ballkontrolle zu bevorzugen.

[6] Beide Varianten wurden auch implementiert und dessen Ergebnisse verglichen. Für mehrere Spiele wurden die Verhalten unter Verwendung beider Varianten ermittelt und miteinander verglichen. Bei der Analyse der unterschiedlich erkannten Verhalten stellte sich heraus, daß die bei der Verwendung der aktiven Ballkontrolle ermittelten Verhalten immer korrekt waren.

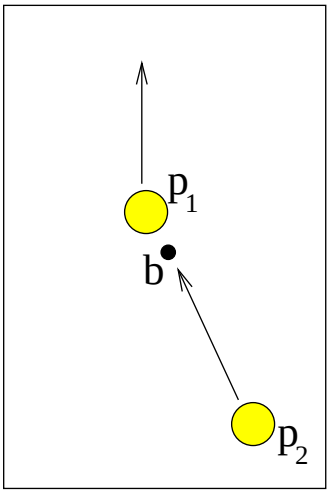


Abbildung 4.3: Beispiel eines Balltransfers der keinen Paß darstellt

eingeführt: der Ball muß sich mindestens **anfänglich** vom Spieler p_1 mit einer **Mindest-Geschwindigkeit**[7] entfernen. Diese Bedingung wird mit dem Ausdruck

$$\text{ballFastDeparting}\,(p_1, t_2, \mathit{ballSpeed}),$$

bezeichnet, wobei t_2 der erste Zeitpunkt ist, bei dem der Kicker den Ball nicht mehr kontrolliert. Die Variable $\mathit{ballSpeed} \in \mathbb{V}$ bezeichnet den Geschwindigkeitsvektor des Balles zum Zeitpunkt t_2.

Desweiteren soll bei einem Paß der Balltransfer von seiten des Kickers gezielt erfolgen, d.h. der Kicker soll sich für einen Empfänger des Passes entscheiden. Leider ist diese Bedingung nicht überprüfbar, da dem MA die Überlegungen der IP verschlossen sind. So könnte es sein, daß der Kicker den Ball einfach nur in eine bestimmte Richtung kicken will ohne die Absicht zu haben, einen Paß auszuführen. Ein Spieler der selben Mannschaft der sich dort aufhält, könnte dann den Ball abfangen und unter seine Kontrolle bringen. Als Beobachter würden wir einen Paß wahrnehmen, da wir dem Kicker die Absicht unterstellen, den Ball gezielt zum Empfänger gekickt zu haben.

Bei einem Paß muß also dem Kicker die Absicht unterstellt werden können, gezielt zu einem Empfänger gekickt zu haben. Eine notwendige Bedingung dafür ist, daß sich mindestens ein **potentieller Empfänger** (beliebiger Spieler[8] des gleichen Teams wie der Kicker) **in dem Bereich** aufhält in den der **Ball gekickt** wurde. Diese Bedingung wird mit

$$\text{teammateInKickRegion}\,(p_1, t_2)$$

bezeichnet, wobei t_2 der erste Zeitpunkt ist, bei dem der Kicker p_1 den Ball nicht mehr kontrolliert. Der besagte Bereich ist dabei durch ein Kreissegment[9] mit

[7] Als Mindest-Geschwindigkeit für den Ball wird ein Wert von 0.95 Meter pro Takt verwendet. Dieser empirisch bestimmte Wert stellt eine gute Abgrenzung für Päße dar.

[8] Der Spieler muß nicht mit dem tatsächlichen Empfänger p_2 übereinstimmen.

[9] Der Radius und der Winkel des Kreissegments sind von der aktuellen Ballgeschwindigkeit abhängig, um so schneller der Ball, um so größer der Radius und so kleiner der Winkel. Die

dem Kicker als Zentrum gegeben. Balltransfers, für die diese Bedingung nicht zutrifft, werden nicht als Pässe angesehen, auch wenn am Ende des Transfers ein Spieler des gleichen Teams den Ball kontrolliert. Solche Transfers werden dann entweder als Klären-Verhalten[10] (Abschnitt 4.4.3 auf Seite 70) oder Dribbel-Verhalten (Abschnitt 4.4.1 auf Seite 67) wahrgenommen.

Bei einem Paß muß der Empfänger im allgemeinen gezielte Bewegungen ausführen um die Kontrolle über den Ball zu erlangen. Wenn der Ball direkt auf den Empfänger zurollt, können diese Bewegungen unter Umständen entfallen (wenn kein Gegner die Möglichkeit hat den Ball vorher abzufangen), dann hält der Empfänger gezielt seine Position. Die **Bewegung des Empfängers** wird mit dem Ausdruck

$$\text{movement}\,(p_2, t_2, t_n, playerMovement)$$

festgehalten, wobei $playerMovement \in \mathbb{V}$ der resultierende Bewegungsvektor des Empfängers p_2 im Zeitraum von der Beendigung der Ballkontrolle durch den Kicker (t_2) bis zur ersten Ballkontrolle durch den Empfänger (t_n) ist. Dieser Ausdruck ist keine notwendige Bedingung zum Erkennen eines Passes, da die Bewegung des Empfängers implizit durch die Differenz seiner Position zum Zeittakt t_2 von seiner Position zum Zeittakt t_n gegeben ist. Mit diesem Ausdruck wird eine weitere Eigenschaft des Passes mit der Variablen $playerMovement$ extrahiert.

Um von einem erfolgreichen Paß sprechen zu können, muß der **Empfänger zum gleichen Team wie** der **Kicker** gehören. Diese Bedingung wird mit dem Ausdruck

$$\text{sameTeam}\,(p_1, p_2)$$

dargestellt. Desweiteren müssen **Empfänger und Kicker unterschiedliche Spieler** sein. Wir kennzeichnen diese Bedingung mit dem Ausdruck

$$\text{notSame}\,(p_1, p_2).$$

Außerdem gibt es einige Umstände, die den Abbruch eines Paß-Verhaltens kennzeichnen. Das Paß-Verhalten wird unter anderem abgebrochen, wenn

- der Ball das Spielfeld verläßt,
- der gegnerische Torwart den Ball fängt, oder
- das Spiel zu Ende ist.

Diese und andere Umstände werden dem Coach durch eine Änderung des Spielzustandes (siehe Abschnitt 3.2 auf Seite 31) mitgeteilt. Pässe finden nur im Spielzustand *play on* (im Spiel) statt, sie können aber auch mit Freistössen etc. begonnen werden. Das bedeutet, daß **außer im ersten Takt** immer der **Spielzustand *play on*** gelten muß, damit von einem Paß gesprochen werden kann. Diese Bedingung wird mit dem Ausdruck

$$\text{virtuallyPlayOn}\,(t_0, t_n),$$

bezeichnet, wobei t_0 der Startzeitpunkt und t_n der Endzeitpunkt des Verhaltens ist. Ein Wechsel des Spielzustandes zum Spielzustand *play on* ist also nur vom ersten zum zweiten Takt erlaubt.

zur Bestimmung von Radius und Winkel benötigten Werte wurden empirisch ermittelt.

[10] Klären im Sinne des Kickens des Balles in den freien Raum

Für ein Paß-Verhalten müssen alle eingeführten Bedingungen gelten. Damit ergibt sich die Definition für das Verhaltens-Muster Paß (Verhaltens-Muster 2).

Verhaltens-Muster 2: Paß von Spieler p_1 zu Spieler p_2

$$\begin{aligned}\text{pass}\,(p_1, p_2, t_0, t_n, ballSpeed, playerMovement) :- \;& \text{XBallControl}_2\,(p_1, t_0, t_1),\\ & \text{ballFree}\,(t_2, t_3),\ \text{XBallControl}_1\,(p_2, t_n, t_n),\ \text{follow}\,(t_1, t_2),\\ & \text{follow}\,(t_3, t_n),\ \text{ballFastDeparting}\,(p_1, t_2, ballSpeed),\\ & \text{teammateInKickRegion}\,(p_1, t_2),\\ & \text{movement}\,(p_2, t_2, t_n, playerMovement),\ \text{sameTeam}\,(p_1, p_2),\\ & \text{notSame}\,(p_1, p_2),\ \text{virtuallyPlayOn}\,(t_0, t_n).\end{aligned}$$

Abbildung 4.4 auf der nächsten Seite zeigt eine Sequenz von Beobachtungen der Zeittakte 150 bis 158. Jede Beobachtung zeigt die Position des Balles und die Positionen der vier Spieler mp_8, mp_9, mp_{11} und op_2 mit ihren jeweiligen Einflußsphären[11]. Die Beobachtungssequenz stellt einen Paß dar. Im folgenden wird die Gültigkeit der verschiedenen Bedingungen des Verhaltens-Musters Paß anhand dieses Beispiels erläutert.

Zum Zeitpunkt 150 (Abb. 4.4(a)) hat der Spieler mp_{11} die alleinige Kontrolle über den Ball, es gilt die Bedingung $\text{XBallControl}_2\,(\text{mp}_{11}, 150, 150)$; zwischen den Zeitpunkten 151 und 157 (Abb. 4.4(b)–(h)) hat kein einziger Spieler die Kontrolle über den Ball, es gilt $\text{ballFree}\,(151, 157)$; und zum Zeitpunkt 158 (Abb. 4.4(i)) kontrolliert der Spieler mp_9 als einziger den Ball, es gilt die Bedingung $\text{XBallControl}_1\,(\text{mp}_9, 158, 158)$. Die follow-Bedingungen gelten, da die Zeitpunkte 150 und 151 bzw. 157 und 158 direkt aufeinander folgen. Der Ball hatte zum Zeitpunkt 151 (Abb. 4.4(b)) eine relativ hohe Geschwindigkeit und der Spieler mp_9 hat sich zu diesem Zeitpunkt auch im Kickbereich von Spieler mp_{11} aufgehalten. Beide Spieler gehören zum selben Team und sind verschieden. Die anderen Bedingungen sind auch erfüllt, es handelt sich also bei der Beobachtungssequenz aus Abbildung 4.4 um ein Paß-Verhalten.

4.2.4 Verschiedene Ausprägungen von Pässen

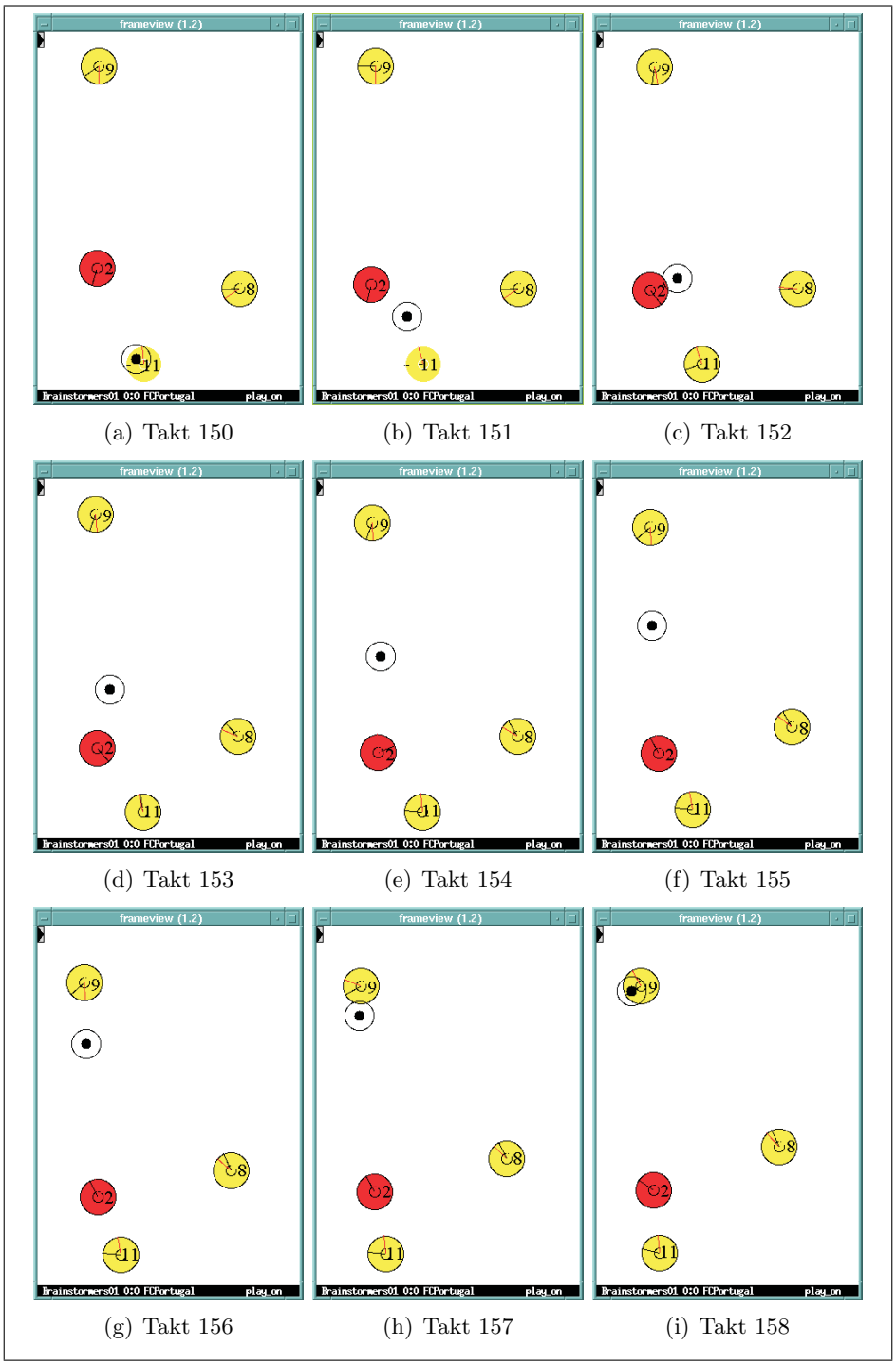
Im letzten Abschnitt wurde das Verhaltens-Muster Paß für einen Kicker (p_1) und einen Empfänger (p_2) von einem Start- (t_0) bis zu einem Ende-Zeitpunkt (t_n) definiert. Neben diesen vier Attributen wurden noch zwei weitere Attribute eingeführt, die einen Paß genauer beschreiben,

ballSpeed **:** Geschwindigkeitsvektor des Balles unmittelbar nach Beendigung der Ballkontrolle durch den Kicker und

playerMovement **:** resultierender Bewegungsvektor des Empfängers im Zeitraum nach Beendigung der Ballkontrolle durch den Kicker bis zur ersten Ballkontrolle durch den Empfänger.

Mit diesen beiden Attributen können verschiedene Ausprägungen vom Paß-Verhalten definiert werden. Einige von diesen Ausprägungen sind in Abbildung 4.5 auf Seite 53 dargestellt.

[11] Befindet sich der Ball im gefärbten Bereich eines Spielers, so ist er in dessen Kick-Bereich.

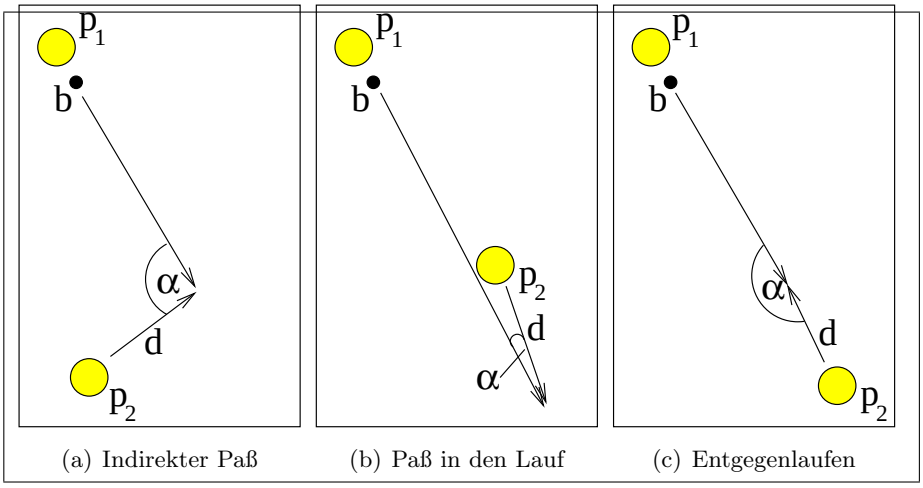


(a) Takt 150 (b) Takt 151 (c) Takt 152

(d) Takt 153 (e) Takt 154 (f) Takt 155

(g) Takt 156 (h) Takt 157 (i) Takt 158

Abbildung 4.4: Aus Beobachtungen der Zeittakte 150 bis 158 zusammengesetzte Beobachtungssequenz, welche dem Verhaltens-Muster Paß entspricht



(a) Indirekter Paß (b) Paß in den Lauf (c) Entgegenlaufen

Abbildung 4.5: Einige Ausprägungen von Pässen

Direktpaß: Bei dieser Ausprägung des Passes führt der Empfänger gar keine oder nur sehr geringe Laufbewegungen durch, da er direkt angespielt wird. Dies wird durch $playerMovement.length \approx 0$ ausgedrückt. Der in Abbildung 4.1 auf Seite 45 dargestellte Balltransfer kann auch als Direktpaß angesehen werden.

Indirekter Paß: Der Gegensatz zum Direktpaß ist der indirekte Paß. Hier führt der Empfänger gezielt Bewegungen aus, um den Ball unter seine Kontrolle zu bringen, es gilt also $playerMovement.length \not\approx 0$. Abbildung 4.5(a) zeigt einen solchen Paß. Indirekte Pässe werden mit den folgenden zwei Ausprägungen noch weiter differenziert.

Paß in den Lauf: Hier wartet der Empfänger des Passes nicht einfach auf den Ball wie beim direkten Paß, sondern läuft eine Weile in die selbe Richtung wie der Ball. Obwohl solch ein Paß vielleicht als Direktpaß vom Kicker geplant war, handelt es sich bei dieser Ausprägung des Passes dennoch um eine Spezialisierung des indirekten Passes. Zusätzlich zu der Bedingung des indirekten Passes gilt hier $\measuredangle(playerMovement, ballSpeed) = \alpha \approx 0$. Ein Paß in den Lauf ist in Abbildung 4.5(b) dargestellt.

Entgegenlaufen: Bei dieser Paßvariante läuft der Empfänger dem Ball entgegen, um ihn möglichst frühzeitig unter seine Kontrolle zu bringen. Wie die Ausprägung „Paß in den Lauf“, handelt es sich beim „Entgegenlaufen“ um einen Spezialfall des indirekten Passes mit der zusätzlichen Bedingung $\measuredangle(playerMovement, ballSpeed) = \alpha \approx \pi$. Abbildung 4.5(c) zeigt eine solche Paßvariante.

Zusätzlich zu diesen Ausprägungen des Paß-Verhaltens, kann man noch weitere Ausprägungen einführen, die durch die Richtung des Passes relativ zum gegnerischen Tor definiert sind. So können u.a. Quer-Pässe, Diagonal-Pässe und Rückpässe unterschieden werden, von denen der Quer-Paß hier vorgestellt wird.

Quer-Paß: Diese Ausprägung des Verhaltens-Muster Paß ist durch eine Bewegung des Balles quer zur Torlinie bestimmt und kann durch die Bedingung $\measuredangle(ballSpeed, goalLine) \approx 0$ ausgedrückt werden, wobei $goalLine \in \mathbb{V}$ den Vektor einer der beiden Torlinien darstellt.

Durch das eingeführte Verhaltens-Muster Paß können nur erfolgreiche Pässe erkannt werden. Die verschiedenen Arten von erfolglosen Pässen (Gegner erlangt Ballkontrolle, Aus, Abseits etc.) werden auch als Ausprägungen des Verhaltens-Muster Paß betrachtet. Ausprägungen bzgl. Erfolglosigkeit sind dabei für viele Verhaltens-Muster identisch. Diese Ausprägungen werden daher separat in Abschnitt 4.4.4 auf Seite 72 eingeführt. Pässe können außerdem danach unterschieden werden, ob sie im Spiel oder bei Freistössen etc. erfolgen. Auch diese Ausprägungen sind für viele Verhalten die gleichen und werden daher auch erst in Abschnitt 4.4.4 untersucht.

Anhand der angegebenen Ausprägungen können Pässe genauer beschrieben werden. Es sind dabei auch Kombinationen von Ausprägungen möglich, so gibt es z.B. Direktpässe die aus einem Freistoß heraus entstehen, oder auch „Pässe in den Lauf“ die gleichzeitig Quer-Pässe sind.

4.3 Formalisierung der Verhaltens-Bestimmung

Der Prozess der Verhaltens-Bestimmung setzt sich zusammen aus:

- der Spezifikation von Verhalten (engl. specify behavior) und
- der Erkennung von Verhalten (engl. recognize behavior).

Diese beiden Teilprozesse sind in Abbildung 4.6 dargestellt und werden in diesem Abschnitt formalisiert. Ein Designer spezifiziert Verhalten durch die Definition von Verhaltens-Mustern. Das Verhaltens-Muster Paß wurde bereits definiert, weitere Verhaltens-Muster werden in Abschnitt 4.4 auf Seite 65 ausgearbeitet. Die Verhaltens-Muster werden zur Erkennung von Verhalten genutzt, die in der virtuellen Fußball-Umgebung auftreten.

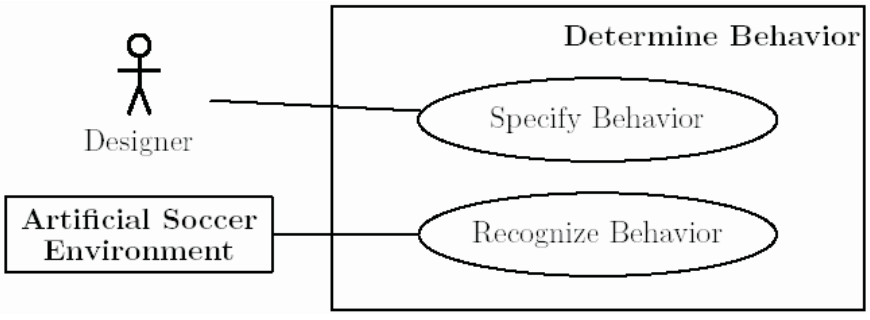


Abbildung 4.6: Struktur der Verhaltens-Bestimmung

Ein Verhalten ist dabei durch eine Sequenz von Ereignissen (bei der Erarbeitung des Verhaltens-Muster durch Bedingungen dargestellt) beschreibbar. Ereignisse und Ereignis-Muster werden daher als Grundbestandteil von Verhalten bzw. Verhaltens-Muster eingeführt (Abschnitt 4.3.1). Bei der Erarbeitung

des Verhaltens-Musters Paß wurden Verhaltens-Ausprägungen vorgestellt. Solche Verhaltens-Ausprägungen werden motiviert (Abschnitt 4.3.2). Dann wird ein Formalismus konzipiert, mit dem Ereignisse und Verhalten spezifiziert werden können (Abschnitt 4.3.3). Schließlich wird gezeigt, wie Ereignisse und Verhalten für gegebene Ereignis-Muster und Verhaltens-Muster erkannt werden (Abschnitt 4.3.4).

4.3.1 Ereignisse als Grundbestandteil von Verhalten

Im letzten Abschnitt wurde das Verhaltens-Muster Paß eingeführt und mehrere notwendige Bedingungen für Pässe aufgestellt. Jede dieser Bedingungen repräsentiert eine Menge von beobachtbaren Ereignissen. So werden zum Beispiel mit der Bedingung $\text{XBallControl}_2\,(p_i, t_j, t_k)$ Ereignisse beschrieben, bei denen ein Spieler p_i für einen auf 2 Takte begrenzten Zeitraum (t_j, t_k) die Kontrolle über den Ball hat.

Ein Ereignis beschreibt dabei eine Beziehung zwischen einzelnen Entitäten oder von Entitäten zur Umgebung. Entitäten in diesem Sinne können Spieler-Agenten, Zeitpunkte oder auch Vektoren sein (also Elemente von $\mathbb{P}$, $\mathbb{T}$ und $\mathbb{V}$). Der Ball und der Spielzustand werden in diesem Kontext als Umgebungsmerkmale betrachtet. Ein Ereignis (z.B. $\text{XBallControl}_2\,(\text{mp}_7, 224, 225)$) besteht aus einem Namen (z.B. XBallControl_2), der eine Beziehung ausdrückt und aus Werten, die die Entitäten der Beziehung darstellen (z.B. $(\text{mp}_7, 224, 225)$). Im Falle des Ereignisses $\text{XBallControl}_2\,(\text{mp}_7, 224, 225)$ wird eine räumliche Beziehung vom Spieler-Agent mp_7 zur Umgebung, genauer zum Ball, beschrieben. Diese Beziehung gilt zu den Zeittakten 224 und 225.

Es können räumliche (z.B. XBallControl_2), zeitliche (z.B. follow) und strukturelle (z.B. sameTeam) Beziehungen unterschieden werden. Ereignisse beschreiben atomare Beziehungen, d.h. sie können nicht weiter zerlegt werden, oder es ist nicht sinnvoll sie weiter zu unterteilen. Ereignisse können dabei sowohl für Zeitpunkte als auch für Zeitintervalle (gegeben durch Start- und Endzeitpunkt) gelten.

Definition 4.1 (Ereignis) *Ein Ereignis ist bestimmt durch einen Namen und einen Tupel von Werten, und beschreibt eine atomare Beziehung von Entitäten zur Umgebung oder von Entitäten untereinander.*

Es wird vorausgesetzt, daß Ereignisse mit dem selben Namen die gleiche Anzahl an Werten haben, und daß die Wertebereiche der Elemente der Tupel von zwei Ereignissen mit dem selben Namen übereinstimmen.

Jede der Bedingungen eines Verhaltens-Muster kann als ein Muster für Ereignisse angesehen werden. Ein solches Muster fasst alle Ereignisse, die den gleichen Namen haben, zusammen. Damit besteht ein Ereignis-Muster aus einem Namen (z.B. XBallControl_2) und mehreren Attributen (z.B. (p_i, t_j, t_k)). Die Attribute stellen dabei beliebige Werte aus den entsprechenden Wertebereichen dar.

Definition 4.2 (Ereignis-Muster) *Ein Ereignis-Muster ist festgelegt durch einen Namen und einen Tupel, welcher aus Attributen für Spieler-Agenten,*

Zeitpunkte und/oder Vektoren besteht. Ein Ereignis-Muster repräsentiert dabei alle Ereignisse des gleichen Namens.

Ereignis-Muster unterscheiden sich von Verhaltens-Mustern dadurch, daß sie atomare Beziehungen zwischen Entitäten beschreiben, wohingegen Verhaltens-Muster komplexere Beziehungen zwischen Entitäten beschreiben. Außerdem setzt sich jedes Verhaltens-Muster aus einer Sequenz von Ereignis-Mustern zusammen. Die Beziehungen zwischen Ereignis-Mustern, Verhaltens-Mustern sowie zwischen Ereignissen und Verhalten sind in Abbildung 4.7 dargestellt.

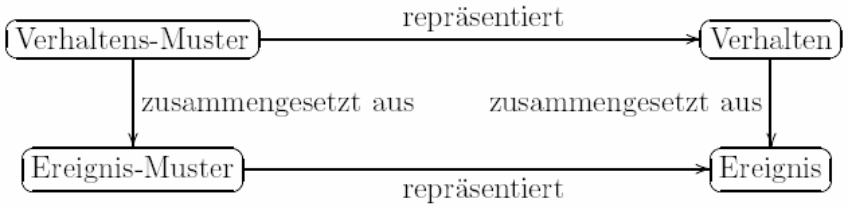


Abbildung 4.7: Beziehungen zwischen Ereignissen, Ereignis-Mustern, Verhalten und Verhaltens-Mustern

Infolge dieser Zusammensetzung gelten Verhaltens-Muster immer für Zeitintervalle mit einem Startzeitpunkt t_0 und einem Endzeitpunkt t_n, wobei $t_n > t_0$ gilt. Ereignis-Muster dagegen können bereits für einzelne Zeitpunkte t_i gelten (darstellbar durch einen gleichen Start- und Endzeitpunkt).

4.3.2 Genauigkeits-Stufen beim Beschreiben von Verhalten

Die genügsamste Art Verhalten zu beschreiben ist es, ihnen Namen (z.B. Paß oder Dribbeln) zuzuordnen, sie also nur durch jeweils einen Namen zu beschreiben. Bei der Definition des Verhaltens-Muster Paß wurden Bedingungen (Ereignis-Muster) eingeführt, die Attribute enthalten. Ein Verhalten ist durch die Belegung seiner Attribute genau bestimmt. Später wurden mit Paß-Ausprägungen die Menge aller Pässe in Kategorien unterteilt und Pässe zusammengefasst, die ähnliche Merkmale aufweisen.

Damit kann ein Verhalten in mehreren Stufen beschrieben werden. Es werden vier Stufen verwendet, die verschiedene Abstraktionsgrade bei der Beschreibung von Verhalten darstellen.

Allgemeine Beschreibung: Hier wird durch einen Namen beschrieben, um welches Verhalten es sich handelt. Beispiele für solche Namen sind: Paß, Dribbeln oder Torschuß.

Ausprägungs-Beschreibung: Auf dieser Stufe wird bestimmt, um welche der Ausprägungen des Verhaltens es sich handelt. Damit werden Untergruppen von Verhalten eingeführt. Bei Pässen wird hier also beschrieben, ob es sich um eine oder mehrere der Paß-Ausprägungen: Direktpaß, Indirekter Paß, Paß in den Lauf oder andere handelt.

Quelle-Ziel-Beschreibung: Für die Beurteilung eines Verhaltens ist die Ausprägungs-Beschreibung meist nicht ausreichend. Aus diesem Grunde werden bei Verhalten, die einen Transfer eines Objektes repräsentieren, Quelle und Ziel des Transfers auf dieser Stufe hinzugenommen. Die Quelle ist dabei bei allen Balltransfers durch den Kicker gegeben. Das Ziel einiger Verhalten ist bereits durch den Namen festgelegt, z.B. Tor beim Torschuß. Bei anderen Balltransfers, wie z.B. Paß, ist das Ziel durch einen Empfänger bestimmt. Eine mögliche Beschreibung auf dieser Stufe könnte sein: „Direktpaß von mp_7 zu mp_5“.

Konkrete Beschreibung: Diese Beschreibung besteht aus allen relevanten Eigenschaften des Verhaltens, d.h. aus der Belegung aller wichtigen Attribute des Verhaltens-Musters. Unwichtige Attribute sind dabei die Zwischenzeitpunkte t_i mit $i \neq 0$, $i \neq n$, von denen abstrahiert wird. Die Attribute des Verhaltens-Muster Paß sind: der Kicker (p_1), der Empfänger (p_2), der Startzeitpunkt (t_0), der Endzeitpunkt (t_n), die Ballgeschwindigkeit ($ballSpeed$) und die Bewegung des Empfängers ($playerMovement$).

Während die allgemeine Beschreibung meist nicht aussagekräftig genug ist und die konkrete Beschreibung in ihrer Gesamtheit durch den Menschen schwer zu verstehen ist, stellen die Zwischenstufen interessante Alternativen zur Beschreibung von Verhalten dar.[12]

Es ist zu erwarten, daß die Genauigkeit der Verhaltens-Vorhersage von der allgemeinen Ebene zur konkreten Ebene abnimmt, wobei anzunehmen ist, daß auf den mittleren Ebenen die Vorhersage ausreichend genau ist. Damit stehen vier verschiedene Ebenen zur Vorhersage von Verhalten zur Verfügung.

Jede Ausprägung eines Verhaltens-Musters (z.B. Ausprägung Quer-Paß oder Ausprägung Direktpaß) kann man auch als eigenes Verhaltens-Muster betrachten. Diese Betrachtungsweise wird bei der Formalisierung der Verhaltensspezifikation (Abschnitt 4.3.3) verwendet. Die unterschiedlichen Ausprägungen eines Verhaltens-Muster (z.B. Paß) weisen jedoch viele Gemeinsamkeiten auf. Bei der objektorientierten Realisierung (Abschnitt 4.5 auf Seite 82) werden sie deshalb zusammengefaßt, um Verhalten effizient erkennen zu können.

4.3.3 Spezifikation von Ereignissen und Verhalten

In diesem Abschnitt werden Ereignisse, Ereignis-Muster, Verhalten und Verhaltens-Muster formal dargestellt. Zunächst werden die Mengen aller Ereignisse und Ereignis-Muster eingeführt. Darauf aufbauend werden die Mengen der Verhalten und Verhaltens-Muster definiert. Abschließend werden die Beziehungen zwischen den eingeführten Ereignissen, Ereignis-Mustern, Verhalten und Verhaltens-Mustern beschrieben.

Zuvor werden noch die Mengen der Namen für Ereignisse, Ereignis-Muster, Verhalten und Verhaltens-Muster eingeführt.

[12] Die beiden mittleren Ebenen können auch in ihrer Reihenfolge vertauscht werden. So entscheidet sich ein Kicker bei einem Paß meist erst für einen Empfänger bevor er sich für eine der Paß-Ausprägungen entscheidet.

$\mathbb{NE}$ ist die Menge der Namen für Ereignisse und Ereignis-Muster. Die Elemente dieser Menge sind dabei beliebige Zeichenfolgen. Einige Namen wurden bereits in Abschnitt 4.2 bei der Erarbeitung des Verhaltens-Muster Paß eingeführt, andere werden in Abschnitt 4.4 bei der Erarbeitung anderer Verhaltens-Muster folgen.

$\mathbb{NB}$ ist die Menge der Namen für Verhalten und Verhaltens-Muster, deren Elemente beliebige Zeichenfolgen sind. Bzgl. der Einführung der Namen gilt das gleiche wie bei der Menge $\mathbb{NE}$.

Die Menge aller Teilmengen einer Menge M, die sogenannte Potenzmenge wird mit 2^M bezeichnet.

4.3.3.1 Ereignisse und Ereignis-Muster

Im letzten Abschnitt wurden Ereignisse durch einen Namen und einen Tupel von Werten definiert. Die Werte können dabei Elemente der Mengen: Spieler-Agenten ($\mathbb{P}$), Zeitpunkte ($\mathbb{T}$), und Vektoren ($\mathbb{V}$) sein. Ereignisse mit verschiedenen Namen haben dabei i.A. eine unterschiedliche Anzahl von Werten mit verschiedenden Wertebereichen. So haben z.B. die Ereignisse mit dem Namen XBallControl_2 3 Werte aus den Wertebereichen $\mathbb{P}$, $\mathbb{T}$, und $\mathbb{T}$, wohingegen die Ereignisse mit dem Namen sameTeam nur 2 Werte aus den Wertebereichen $\mathbb{P}$ und $\mathbb{P}$ haben. Die für diese Arbeit benötigten Ereignisse bestehen aus

- einem Namen $name \in \mathbb{NE}$,
- keinem, einem, oder zwei Spieler-Agenten $p \in \mathbb{P}$ bzw. keiner, einer, oder zwei Mengen von Spieler-Agenten[13] $P \in 2^{\mathbb{P}}$,
- keinem, einem, oder zwei Zeitpunkten $t \in \mathbb{T}$, und
- keinem oder einem Vektor $v \in \mathbb{V}$.

Dabei muß ein Ereignis mindestens einen Wert neben dem Namen aufweisen.

Da Ereignisse mit dem selben Namen die gleiche Anzahl an Werten mit gleichen Wertebereichen haben sollen (siehe Bemerkung im letzten Abschnitt unter Definition 4.1 auf Seite 55), werden zuerst die Mengen aller Ereignis-Wertetupel mit dem Namen $name \in \mathbb{NE}$ eingeführt. Darauf aufbauend werden die Menge aller Ereignisse und die Menge aller Ereignis-Muster definiert.

$\texttt{eventTupel}^{name}$ bezeichnet die Menge aller **Ereignis-Wertetupel** mit dem Namen $name \in \mathbb{NE}$. Dabei gilt

$$\begin{aligned} &\texttt{eventTupel}^{name} \subseteq \mathbb{P}^i \times \left(2^{\mathbb{P}}\right)^j \times \mathbb{T}^k \times \mathbb{V}^l \\ &\text{wobei } i,j,k \in \{0,1,2\}, l \in \{0,1\}, i+j+k+l \geq 1, i = 0 \vee j = 0. \end{aligned} \tag{4.1}$$

Ein Index i im Exponent einer Menge stellt dabei das i-fache Kreuzprodukt dieser Menge dar.

[13] Mengen von Spieler-Agenten werden z.B. bei den Ereignissen des Abseits-Verhaltens benötigt.

Mit den Wertebereichen für die Exponenten i, j, k, und l werden die unterschiedlichen Bestandteile aus der obigen Aufzählung für die Ereignisse bereitgestellt. Die Bedingung $i + j + k + l \geq 1$ stellt sicher, daß jedes Ereignis-Wertetupel mindestens einen Wert aufweist. Die gewünschte Ausschließlichkeit von Spieler-Agenten und Mengen von Spieler-Agenten für Ereignisse wird durch den Ausdruck $i = 0 \vee j = 0$ erreicht.

Für jeden Namen *name* haben die Indizes i, j, k, und l bestimmte Werte, d.h. die Indizes sind durch den Namen definiert. Alle Ereignis-Wertetupel $valueSeq \in \texttt{eventTupel}^{name}$ haben damit die gleiche Anzahl an Werten und die gleichen Wertebereiche.

Jede Ereignis-Wertetupel-Menge $\texttt{eventTupel}^{name}$ ist dabei eine Teilmenge des Kreuzproduktes $\mathbb{P}^i \times \left(2^{\mathbb{P}}\right)^j \times \mathbb{T}^k \times \mathbb{V}^l$. Die Teilmenge ist durch den Namen (die Bedeutung) der Ereignis-Wertetupel-Menge bestimmt. So enthält die Menge der Ereignis-Wertetupel $\texttt{eventTupel}^{sameTeam}$ nur die Paare von $\mathbb{P} \times \mathbb{P}$, bei denen beide Spieler zum selben Team gehören.

Für die Menge der Ereignis-Wertetupel $\texttt{eventTupel}^{\text{ballFree}}$ haben die Exponenten i, j, und l den Wert 0 und der Exponent k den Wert 2, es gilt also $\texttt{eventTupel}^{\text{ballFree}} \subseteq \mathbb{T}^2 = \mathbb{T} \times \mathbb{T}$. Damit ist jedes Ereignis-Wertetupel $valueSeq \in \texttt{eventTupel}^{\text{ballFree}}$ ein Element von $\mathbb{T} \times \mathbb{T}$. Die Ereignis-Wertetupel-Menge $\texttt{eventTupel}^{\text{ballFree}}$ ist dabei die Teilmenge von $\mathbb{T} \times \mathbb{T}$, die alle Zeitpunkt-Paare enthält, bei denen der erste Zeitpunkt kleiner oder gleich dem zweiten Zeitpunkt ist.

`events` bezeichnet die Menge aller **Ereignisse**. Ein Ereignis besteht aus einem Namen und einem Tupel von Werten (vgl. Definition 4.1 auf Seite 55). Der Name ist ein Element der Menge $\mathbb{NE}$. Der Wertetupel ist durch ein Element der entsprechenden Ereignis-Wertetupel-Menge $\texttt{eventTupel}^{name}$ bestimmt.

$$\texttt{events} = \big\{ name\,(valueSeq) : name \in \mathbb{NE}, \; valueSeq \in \texttt{eventTupel}^{name} \big\} \tag{4.2}$$

Damit ist z.B. $\text{XBallControl}_2\,(\text{mp}_7, 224, 225)$ ein Ereignis aus der Menge `events`.

`eventPatterns` bezeichnet die Menge aller **Ereignis-Muster**. Ein Ereignis-Muster $EP \in \texttt{eventPatterns}$ repräsentiert alle Ereignisse gleichen Namens (vgl. Definition 4.2 auf Seite 56). Dies gilt für Ereignis-Muster $EP = name\,(\texttt{eventTupel}^{name})$ mit $name \in \mathbb{NE}$. Damit ergibt sich die Menge aller Ereignis-Muster als

$$\texttt{eventPatterns} = \big\{ name\,(\texttt{eventTupel}^{name}) : name \in \mathbb{NE} \big\}. \tag{4.3}$$

Das Ereignis-Muster $EP = \text{ballFree}\,(\texttt{eventTupel}^{\text{ballFree}})$ repräsentiert beispielsweise alle Ereignisse $E = \text{ballFree}\,(valueSeq)$ mit $valueSeq \in \texttt{eventTupel}^{\text{ballFree}} \subseteq \mathbb{T} \times \mathbb{T}$.

Ein Ereignis-Muster kann ebenso mit Attributen beschrieben werden. Wir verwenden deshalb auch die Attributsbezeichnung und benennen z.B. das Ereignis-Muster $\text{ballFree}\,(\texttt{eventTupel}^{\text{ballFree}})$ auch mit $\text{ballFree}\,(t_i, t_j)$.

Die Mengen für Ereignisse und Ereignis-Muster wurden aufbauend auf den Mengen der Ereignis-Wertetupel definiert.

$$\begin{aligned}\texttt{events} =& \{name\,(valueSeq) : name \in \mathbb{NE},\\ & valueSeq \in \texttt{eventTupel}^{name}\}\\ \texttt{eventPatterns} =& \{name\,(\texttt{eventTupel}^{name}) : name \in \mathbb{NE}\}\end{aligned}$$

Damit beschreibt jedes Ereignis-Muster $EP = name\,(\texttt{eventTupel}^{name})$ eine Ereignismenge $\{E_1, \ldots E_n\}$ mit $E_i = name\,(valueSeq_i)$ und $valueSeq_i \in \texttt{eventTupel}^{name}$. Jedes dieser Ereignisse E_i wird durch das Ereignis-Muster EP repräsentiert. Diese Beziehung wird im folgenden mit $E_i \Subset EP$ gekennzeichnet.

4.3.3.2 Verhalten und Verhaltens-Muster

Wie schon das Verhaltens-Muster Paß durch eine Sequenz von Bedingungen und damit durch eine Sequenz von Ereignis-Mustern festgelegt wurde (siehe Verhaltens-Muster 2 auf Seite 51 und Abschnitt 4.3.1 auf Seite 55), werden auch alle anderen Verhaltens-Muster durch eine Sequenz von Ereignis-Mustern definiert.

Zwischen den Attributen der Ereignis-Muster eines Passes wurden einige Variablenbindungen durchgeführt. Diese müssen formal abgebildet werden. Dazu werden Unifikatoren definiert, wobei ein Unifikator zwei bestimmte Attribute zweier Ereignis-Muster miteinander bindet. Danach führen wir die Verhaltensmengen analog den Ereignismengen ein, d.h. wir beginnnen mit der Definition der Mengen aller Verhaltens-Wertetupel mit dem Namen $name \in \mathbb{NB}$. Ausgehend von dieser Definition wird die Menge aller Verhalten und die Menge aller Verhaltens-Muster eingeführt.

`uni` ist die Menge aller **Unifikatoren**, wobei ein Unifikator $u \in \texttt{uni}$ ein Quadrupel natürlicher Zahlen ist, $u = (i, j, k, l)$ mit $i, j, k, l \in \mathbb{N}$.

$$\texttt{uni} = \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \tag{4.4}$$

Die Unifikatoren werden zur Variablenbindung zwischen den einzelnen Ereignis-Mustern $EP_h \in \texttt{eventPatterns}$ $(h = 1, \ldots, n)$ einer Ereignis-Muster-Sequenz $\overline{EP} = (EP_1, \ldots EP_n)$ genutzt. Dabei bedeutet ein Unifikator $u = (i, j, k, l)$, daß die j-te Variable von Ereignis-Muster EP_i mit der l-ten Variable von Ereignis-Muster EP_k gebunden ist, d.h. die entsprechenden Variablenwerte müssen identisch sein. Eine Menge von Unifikatoren $U \subset \texttt{uni}$ definiert eine Menge von solchen Variablenbindungen.

$\texttt{behaviorTupel}^{name}$ bezeichnet die Menge aller **Verhaltens-Wertetupel** mit dem Namen $name \in \mathbb{NB}$. Die Verhaltens-Wertetupel-Menge ist durch eine Sequenz von Ereignis-Mustern $\overline{EP} = (EP_1, \ldots EP_n) \in \texttt{eventPatterns}^n$ und durch eine Menge von Unifikatoren $U \subset \texttt{uni}$ definiert. Die Sequenz (inklusive dessen Stelligkeit) und die Menge sind dabei durch den Namen

name eindeutig festgelegt. $\overline{EP}$ wird als Ereignis-Muster-Sequenz und U als Unifikator-Menge von $\texttt{behaviorTupel}^{name}$ bezeichnet. Es gilt

$$\begin{aligned}\texttt{behaviorTupel}^{name} = \Big\{&(E_1,\ldots,E_n) : E_h \Subset EP_h, h = 1,\ldots,n,\\ &\big(\forall(i,j,k,l) \in U : E_i.j = E_k.l\big)\Big\},\\ \text{mit fixierten } \overline{EP} = (EP_1,\ldots EP_n) &\in \texttt{eventPatterns}^n, U \subset \texttt{uni}.\end{aligned} \tag{4.5}$$

Der Ausdruck $E_i.j$ bezeichnet dabei das j-te Element v_j des Ereignis-Wertetupels von Ereignis $E_i = name_i(v_1,\ldots,v_m)$.

Ein Verhaltens-Wertetupel $eventSeq \in \texttt{behaviorTupel}^{name}$ besteht damit aus einer Sequenz von Ereignissen $eventSeq = \overline{E} = (E_1,\ldots,E_n)$. Jedes Ereignis E_h $(h = 1,\ldots,n)$ wird dabei durch ein entsprechendes Ereignis-Muster EP_h repräsentiert, $E_h \Subset EP_h$.

Durch die Unifikatoren $(i,j,k,l) \in U$ wird die Menge aller Ereignis-Sequenzen $\{(E_1,\ldots,E_n) : E_h \Subset EP_h, h = 1,\ldots,n\}$ auf eine Teilmenge eingeschränkt.

Die Verhaltens-Wertetupel-Menge $\texttt{behaviorTupel}^{\text{ballTransfer}}$ ist beispielsweise durch die Sequenz von Ereignis-Mustern $\overline{EP^{\text{ballTransfer}}}$ und durch die Unifikatormenge $U^{\text{ballTransfer}}$ definiert.

$$\begin{aligned}\overline{EP^{\text{ballTransfer}}} &= \big(\text{XBallControl}_2\,(p_1,t_0,t_1), \text{ballFree}\,(t_2,t_3),\\ &\qquad \text{XBallControl}_1\,(p_2,t_n,t_n), \text{follow}\,(t_i,t_j), \text{follow}\,(t_k,t_l)\big)\\ U^{\text{ballTransfer}} &= \big\{(1,3,4,1),(2,1,4,2),(2,2,5,1),(3,2,5,2)\big\}\end{aligned}$$

Mit der Unifikatormenge $U^{\text{ballTransfer}}$ werden die benötigten Variablenbindungen $t_1 = t_i, t_2 = t_j, t_3 = t_k, t_n = t_l$ realisiert. Damit fehlt nur noch der Name, damit die Verhaltens-Wertetupel-Menge $\texttt{behaviorTupel}^{\text{ballTransfer}}$ dem Verhaltens-Muster Balltransfer (Verhaltens-Muster 1 auf Seite 46) entspricht.

`behaviors` bezeichnet die Menge aller **Verhalten**. Jedes Verhalten besteht aus einem Namen $name \in \mathbb{NB}$ und einer Sequenz von Ereignissen $eventSeq \in \texttt{behaviorTupel}^{name}$.

$$\begin{aligned}\texttt{behaviors} = \big\{&name\,(eventSeq) : name \in \mathbb{NB},\\ &eventSeq \in \texttt{behaviorTupel}^{name}\big\}\end{aligned} \tag{4.6}$$

Ein Beispiel für ein Verhalten B des Verhaltens-Muster Balltransfer ist

$$\begin{aligned}B = \text{ballTransfer}\big(&\text{XBallControl}_2\,(\text{mp}_7,224,225), \text{ballFree}\,(226,231),\\ &\text{XBallControl}_1\,(\text{mp}_9,232,232), \text{follow}\,(225,226),\\ &\text{follow}\,(231,232)\big)\end{aligned}$$

`behaviorPatterns` bezeichnet die Menge aller **Verhaltens-Muster**. Alle Verhalten mit den gleichen Namen werden durch dasselbe Verhaltens-Muster $BP \in$ `behaviorPatterns` repräsentiert (vgl. Definition 3.6 auf Seite 35). Dies gilt für Verhaltens-Muster $BP = name\,(\texttt{behaviorTupel}^{name})$ mit $name \in \mathbb{NB}$. Daraus resultiert die Menge aller Verhaltens-Muster

$$\texttt{behaviorPatterns} = \big\{name\,(\texttt{behaviorTupel}^{name}) : name \in \mathbb{NB}\big\}. \tag{4.7}$$

Mit $BP = \text{ballTransfer}\,(\texttt{behaviorTupel}^{\text{ballTransfer}})$ ist also formal das Verhaltens-Muster Balltransfer beschrieben.

Die beiden Mengen für Verhalten und für Verhaltens-Muster wurden aufbauend auf den Mengen der Verhaltens-Wertetupel eingeführt.

$$\begin{aligned}\texttt{behaviors} =& \big\{name\,(eventSeq) : name \in \mathbb{NB},\\ & \quad eventSeq \in \texttt{behaviorTupel}^{name}\big\}\\ \texttt{behaviorPatterns} =& \big\{name\,(\texttt{behaviorTupel}^{name}) : name \in \mathbb{NB}\big\}\end{aligned}$$

Jedes Verhaltens-Muster $BP = name\,(\texttt{behaviorTupel}^{name})$ beschreibt eine Verhaltensmenge $\{B_1, \ldots, B_m\}$ mit $B_i = name\,(eventSeq_i)$ und $eventSeq_i \in \texttt{behaviorTupel}^{name}$. Damit ist jedes dieser Verhalten B_i durch das Verhaltens-Muster BP repräsentiert. Diese Beziehung zwischen B_i und BP wird im folgenden mit $B_i \Subset BP$ gekennzeichnet.

4.3.3.3 Beziehungen zwischen Ereignissen und Verhalten sowie ihrer Muster

In den letzten beiden Abschnitten wurden die Mengen der Ereignisse, der Ereignis-Muster, der Verhalten, und der Verhaltens-Muster aufeinander aufbauend definiert. Diese Definitionen erfolgten so, daß: Ereignisse ($E = name_e(\ldots)$, $name_e \in \mathbb{NE}$) durch Ereignis-Muster ($EP = name_e(\ldots)$) repräsentiert sind ($E \Subset EP$), Verhaltens-Muster ($BP = name_b(\ldots)$, $name_b \in \mathbb{NB}$) u.a. durch eine Sequenz von Ereignis-Mustern definiert sind (BP defniert durch $\overline{EP} = (EP_1, \ldots EP_n)$), und Verhalten ($B = name_b(\ldots)$) einerseits durch Verhaltens-Muster repräsentiert sind ($B \Subset BP$) und andererseits eine Sequenz von Ereignissen darstellen ($B = name_b(E_1, \ldots E_n)$).

Die Beziehungen zwischen diesen vier Mengen sind in Abbildung 4.8 auf der nächsten Seite als ERM (Entity-Relationship-Model) wiedergegeben. Im oberen Teil jeder ovalen Box (Entity) wird ein Beispiel-Element gezeigt. Im unteren Teil wird die Beziehung dieses Element zu den Elementen der anderen Mengen wiedergegeben. Die zahlenmäßige Beziehung wird durch das Relationssymbol $\Diamond$ und durch die Kardinalitäten (a, b) dargestellt. Folgende zahlenmäßige Beziehungen gelten zwischen den vier Mengen:

- 1 Ereignis-Muster repräsentiert 0 bis m Ereignisse $((0, *))$,
- 1 Verhaltens-Muster wird durch 2 bis n Ereignis-Muster definiert $((2, *))$,

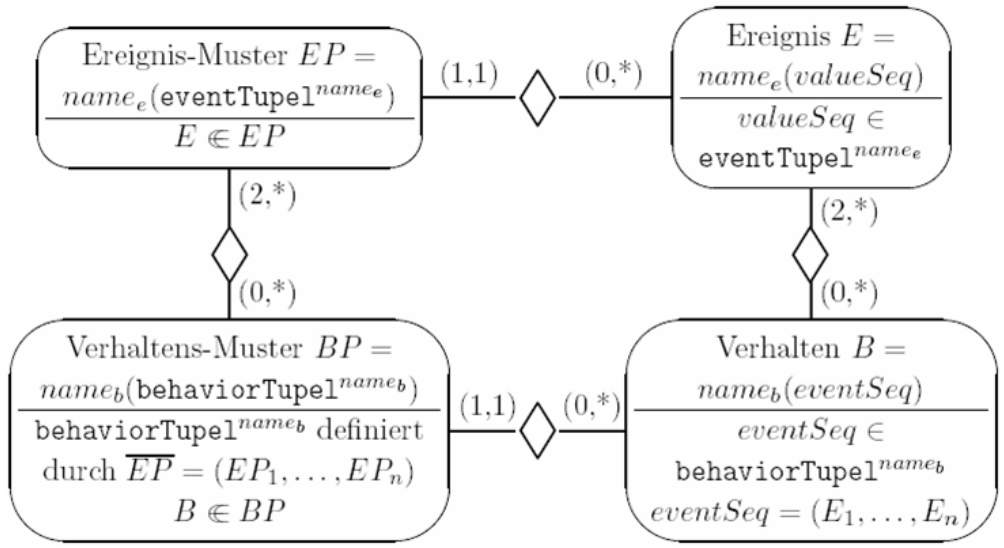


Abbildung 4.8: Beziehungen zwischen Ereignissen, Ereignis-Mustern, Verhalten und Verhaltens-Mustern als ER-Model

- 1 Verhaltens-Muster repräsentiert 0 bis k Verhalten $((0, *))$,
- 1 Verhalten besteht aus 2 bis n Ereignissen $((2, *))$,
- 1 Ereignis wird durch genau ein Ereignis-Muster repräsentiert $((1, 1))$,
- 1 Ereignis-Muster kann an der Definition von 0 bis p Verhaltens-Muster beteiligt sein $((0, *))$,
- 1 Verhalten wird durch genau ein Verhaltens-Muster repräsentiert $((1, 1))$, und
- 1 Ereignis kann zu 0 bis q Verhalten gehören $((0, *))$.

Prinzipiell ist es auch möglich Ereignis-Muster bzw. Verhaltens-Muster zu definieren, die keine Ereignisse bzw. Verhalten repräsentieren. Dies kann durch die Erzeugung eines Widerspruches bei der Definition passieren. Ereignis-Muster bzw. Verhaltens-Muster sind allerdings nur sinnvoll, wenn sie mindestens ein Ereignis bzw. Verhalten repräsentieren.

Für die zahlenmäßigen Beziehungen lassen sich leicht Beispiele angeben. So ist z.B. das Verhaltens-Muster ballTransfer durch 5 Ereignis-Muster definiert. Das Ereignis-Muster XBallControl_2 ist an der Definition mehrerer Verhaltens-Muster wie z.B. Paß und BallTransfer beteiligt.

4.3.4 Erkennung von Verhalten

Zur Erkennung von Verhalten wird eine Beobachtungssequenz mit einem Verhaltens-Muster darauf verglichen, ob sie zusammenpassen, ein sogenanntes *Pattern Matching* wird durchgeführt. Wenn das Verhaltens-Muster und ein Teil der Beobachtungssequenz zusammenpassen, wird ein Verhalten erkannt. Dabei

werden die Werte der Attribute innerhalb eines Verhaltens-Musters, d.h. ein konkretes Verhalten bestimmt. Beim Vergleich der in Abbildung 4.4 auf Seite 52 dargestellten Beobachtungssequenz mit dem Verhaltens-Muster Paß wird ein Paß-Verhalten erkannt und die Attribute des Passes, z.B. Kicker (p_1) und Empfänger (p_2), werden bestimmt ($p_1 = \mathrm{mp}_{11}$, $p_2 = \mathrm{mp}_9$).

Der Prozess der Verhaltens-Erkennung ist zweigeteilt. Im ersten Schritt werden Ereignisse wahrgenommen, indem die Beobachtungssequenzen mit Ereignis-Mustern verglichen werden ($\mathrm{recognize_E}$). Im zweiten Schritt werden Verhalten wahrgenommen, indem die im ersten Schritt wahrgenommenen Ereignisse mit den Verhaltens-Mustern verglichen werden ($\mathrm{recognize_B}$). Aus diesen zwei Schritten setzt sich der Wahrnehmungsprozess von Verhalten zusammen (recognize).

`observations`* ist die Menge der Beobachtungssequenzen, wobei eine Beobachtungssequenz $O^*(t_i, t_j) \in$ `observations`* eine Sequenz von aufeinanderfolgenden Beobachtungen $O_{t_i}, \ldots, O_{t_j}$ ist.

$$\begin{aligned}\texttt{observations}^* = \big\{&(O_{t_1}, \ldots, O_{t_n}) : O_{t_i} \in \texttt{observations},\\ &(\forall t_i, t_{i+1} : t_{i+1} = t_i + 1), n \geq 2\big\}\end{aligned} \tag{4.8}$$

Zu kurze Beobachtungssequenzen sind zur Erkennung von Verhalten nicht geeignet, da für solche selten vollständige Verhalten wahrgenommen werden können. Für die Analyse werden ganze Spiele verwendet, d.h. im Normalfall die Beobachtungssequenz $O^*(0, 6000)$.

$\mathrm{recognize_E}$ ist die Funktion mit der Ereignisse erkannt werden. Für eine Beobachtungssequenz $O^* \in$ `observations`* und ein Ereignis-Muster $EP \in$ `eventPatterns` werden alle Ereignisse, die durch das Ereignis-Muster repräsentiert sind und in der Beobachtungssequenz gelten, bestimmt. Die erkannte Ereignismenge kann dabei auch leer sein.

$$\begin{aligned}&\mathrm{recognize_E} : \texttt{observations}^* \times \texttt{eventPatterns} \rightarrow 2^{\texttt{events}}\\ &\mathrm{recognize_E}\big(O^*, EP\big) = \big\{E_1, \ldots, E_n\big\} \quad \text{mit}\\ &\qquad\qquad E_i \Subset EP, i = 1, \ldots, n\end{aligned} \tag{4.9}$$

Für die in Abbildung 4.4 auf Seite 52 dargestellte Beobachtungssequenz können z.B. zwei Ereignisse für das Ereignis-Muster $\mathrm{XBallControl}_1(\ldots)$ bestimmt werden. Diese Ereignisse sind $\mathrm{XBallControl}_1(\mathrm{mp}_{11}, 150, 150)$ und $\mathrm{XBallControl}_1(\mathrm{mp}_9, 158, 158)$.

$\mathrm{recognize_B}$ ist die Funktion zum Erkennen von Verhalten. Dabei werden ausgehend von einer Menge von Ereignissen alle Verhalten für ein Verhaltens-Muster bestimmt. Im allgemeinen können für ein Verhaltens-Muster mehrere Verhalten erkannt werden.

$$\begin{aligned}&\mathrm{recognize_B} : 2^{\texttt{events}} \times \texttt{behaviorPatterns} \rightarrow 2^{\texttt{behaviors}}\\ &\mathrm{recognize_B}\big(\{E_1, \ldots, E_m\}, BP\big) = \big\{B_1, \ldots, B_n\big\} \quad \text{mit}\\ &\qquad\qquad B_i \Subset BP, i = 1, \ldots, n\end{aligned} \tag{4.10}$$

recognize ist die resultierende Funktion zur Verhaltenserkennung. Sie ist aus den beiden Funktionen $\text{recognize}_\text{E}$ und $\text{recognize}_\text{B}$ zusammengesetzt und bestimmt für ein Verhaltens-Muster $BP = name\,(\texttt{behaviorTupel}^{name})$ alle Verhalten, die in einer Beobachtungssequenz O^* gelten.

Das Verhaltens-Muster BP ist durch eine Ereignis-Muster-Sequenz $\overline{EP} = (EP_1, \ldots EP_n)$ und durch einen Unifikator $U \subset \texttt{uni}$ definiert (vgl. Formel 4.5 und Formel 4.7). Die Funktion $\text{recognize}_\text{E}$ wird für alle Ereignis-Muster EP_i dieser Ereignis-Muster-Sequenz angewendet. Auf die Vereinigung der daraus resultierenden Ereignismengen wird danach die Funktion $\text{recognize}_\text{B}$ für das Verhaltens-Muster BP angewendet. Auf diese Weise werden alle Verhalten des Verhaltens-Muster BP innerhalb der Beobachtungssequenz O^* erkannt.

$$\text{recognize} : \texttt{observations}^* \times \texttt{behaviorPatterns} \rightarrow 2^{\texttt{behaviors}}$$
$$\text{recognize}\big(O^*, BP\big) = \text{recognize}_\text{B}\Big(\Big(\bigcup_{1\leq i\leq n} \text{recognize}_\text{E}\big(O^*, EP_i\big)\Big), BP\Big) \tag{4.11}$$

mit $\overline{EP} = (EP_1, \ldots EP_n)$ ist die Ereignis-Muster-Sequenz von BP

Für die Beobachtungssequenz, die in Abbildung 4.4 auf Seite 52 dargestellt ist, kann mit dieser Funktion für das Verhaltens-Muster Paß genau ein Paß erkannt werden.

In Abbildung 4.9 ist die Verhaltens-Erkennung graphisch zusammengefasst. Ereignisse werden durch die Funktion $\text{recognize}_\text{E}$ erkannt, die dafür Ereignis-Muster und Beobachtungssequenzen nutzt. Die Funktion $\text{recognize}_\text{B}$ nutzt danach die wahrgenommenen Ereignisse mehrerer Ereignis-Muster, um die in der Beobachtungssequenz geltenden Verhalten für gegebene Verhaltens-Muster zu bestimmen.

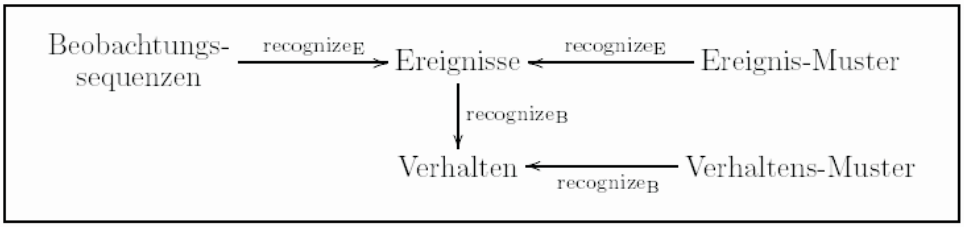


Abbildung 4.9: Der Prozess der Verhaltens-Erkennung

4.4 Weitere Verhaltens-Muster

Das Verhaltens-Muster Paß wurde bereits in Abschnitt 4.2 vorgestellt. In diesem Abschnitt werden weitere Verhaltens-Muster eingeführt. Wir beschränken uns dabei auf Verhaltens-Muster, die direkt Einfluß auf die Bewegung des Balles nehmen. Für diese Verhaltens-Muster läßt sich der Start und das Ende eines

Verhaltens relativ leicht festlegen. Bei Verhalten ohne Ball ist dies mitunter sehr schwierig. Der in Kapitel 3 beschriebene Modellierungsansatz wird für die folgenden vier Verhaltens-Muster, bei deren Start jeweils genau ein Spieler den Ball kontrolliert, komplett realisiert:

Paß: Zielgerichteter Transfer des Balles zu einem Mitspieler des selben Teams (Abschnitt 4.2)

Dribbeln: Bewegen des Balles unter ständiger Beibehaltung der Ballkontrolle bzw. mit kurzzeitiger Aufgabe der Ballkontrolle (Abschnitt 4.4.1)

Torschuß: Gezielter Schuß auf das gegnerische Tor (Abschnitt 4.4.2)

Klären: Weiträumige Beförderung des Balles in den freien Raum durch Kicken des Balles (Abschnitt 4.4.3)

Bei jedem dieser vier Verhaltens-Muster kann es zu Fehlern kommen, die durch den Kontrollverlust des Balles an das gegnerische Team bestimmt sind. Dabei treten bei diesen Verhaltens-Mustern die gleichen Ausprägungen von Fehlern auf. Die **Fehler-Ausprägungen** dieser Verhaltens-Muster werden daher gesondert eingeführt (Abschnitt 4.4.4). Desweiteren kann der Start jedes dieser vier Verhalten im Spiel oder bei einem Freistoß etc. erfolgen. Auch die **Start-Ausprägungen** werden separat behandelt (Abschnitt 4.4.4).

Die **Gemeinsamkeiten und Unterschiede** der Verhaltens-Muster Paß, Dribbeln, Torschuß, und Klären werden zusammengefasst (Abschnitt 4.4.5). Dabei wird auch untersucht, inwieweit durch diese vier Verhaltens-Muster alle möglichen Verhalten abgedeckt sind, an deren Start und Ende genau ein Spieler den Ball kontrolliert.

Um die weiteren Möglichkeiten des Ansatzes zu zeigen, werden außerdem noch zwei weitere Verhalten betrachtet, für die aber nur die Verhaltens-Muster definiert werden.

Doppelpaß: Zusammenspiel zweier Spieler des selben Teams, wobei der den Ball kontrollierende Spieler den Ball zum anderen Spieler passt, der dann sofort den Ball zum ersten Spieler zurückpasst. Ziel dieses Manövers ist es, die gegnerische Verteidigung auszumanövrieren (Abschnitt 4.4.6).

Abseitsfalle: Koordiniertes Verhalten einer Menge von Spielern eines Teams mit dem Ziel einen oder mehrere gegnerische Spieler in eine Abseits-Position zu bringen (Abschnitt 4.4.7).

Damit werden in diesem Abschnitt fünf neue Verhaltens-Muster jeweils durch eine Sequenz von Ereignis-Mustern (auch mit Bedingungen bezeichnet) definiert. In den folgenden Ausführungen wird also geklärt, **was** die obigen Verhaltens-Muster bedeuten und **wie** sie beschrieben werden können. Die Frage **warum** bzw. **wann** ein Agent bzw. eine Gruppe von Agenten Verhalten der Verhaltens-Muster Paß, Dribbeln, Torschuß und Klären ausführen wird später untersucht (Abschnitt 5.1).

4.4.1 Das Verhaltens-Muster Dribbeln

Allgemein wird unter Dribbeln die Bewegung des Balles durch einen Spieler (im folgenden auch Dribbler genannt) unter ständiger Beibehaltung der Ballkontrolle angesehen. Eine Bewegung des Dribblers soll dabei nicht notwendig sein, d.h. es werden auch Verhalten bei denen sich der Spieler nicht bewegt, als Dribbeln interpretiert.

Im verwendeten Szenario ist klar definiert was Ballkontrolle bedeutet (siehe Abschnitt 4.2.2 auf Seite 46) und damit auch, was ständige Ballkontrolle bedeutet (Ballkontrolle zu jedem Zeittakt). Bei der Beobachtung eines Spiels wird ständige Ballkontrolle jedoch meist nicht so eingeschränkt gesehen. Ein Beobachter akzeptiert bei einem Dribbler meist auch eine kurzzeitige Aufgabe der Ballkontrolle. Ein menschlicher Beobachter bei einem Fußballspiel kann oft auch nicht sicher entscheiden ob ein Spieler den Ball kontrolliert oder nicht.

Zuerst wird die Ausprägung des Verhaltens-Muster Dribbeln untersucht, bei der der Dribbler die ständige Kontrolle über den Ball hat. Danach geht es um die Dribbel-Ausprägung bei der der Dribbler vorübergehend die Kontrolle über den Ball aufgibt. Schließlich werden weitere Ausprägungen des Dribbel-Verhaltens vorgestellt.

4.4.1.1 Dribbeln mit ständiger Ballkontrolle

Für das Dribbeln mit ständiger Ballkontrolle muß verständlicherweise der dribbelnde Spieler über den ganzen Zeitraum (d.h. in jedem Zeittakt des Zeitraums) die Kontrolle über den Ball haben. Desweiteren darf in diesem Zeitraum kein anderer Spieler den Ball kontrollieren. Der Ausdruck

$$\text{XballControl}\,(p_1, t_0, t_n)$$

drückt die **alleinige Ballkontrolle** des Spielers p_1 im Zeitraum t_0 bis t_n aus. Ein Dribbel-Verhalten ist also beendet, wenn der obige Ausdruck nicht mehr gilt.

Beim Start und beim Ende eines Passes kam die Bedingung XballControl mit der Einschränkung auf zwei Zeittakte bzw. einen Zeittakt (XBallControl_2, XBallControl_1) ebenso vor. Damit solche kurzfristigen Ballkontrollen nicht als Dribbel-Verhalten angesehen werden, wird die Bedingung eingeführt, daß ein Dribbel-Verhalten eine Dauer von **mindestens 4 Takten**[14] haben muß. Diese Bedingung wird mit

$$\text{minTime}_4\,(t_0, t_n),$$

bezeichnet, mit t_0 als Start- und t_n als Endzeitpunkt des Dribbel-Verhaltens.

Es ist möglich, daß ein Spieler einen sehr langen Zeitraum dribbelt (den Ball kontrolliert). Mit den bisherigen Bedingungen würde dies als ein Dribbel-Verhalten erkannt werden, welches sich im Extremfall auch über das ganze Spiel erstrecken könnte. Dabei würde der Dribbler aber des öfteren seine Laufrichtung ändern. Für diese Änderung der Laufrichtung gibt es irgendwelche

[14] Die Wahl von 4 Takten hat sich empirisch als gut herausgestellt. Bei der Verwendung von 3 Takten wurde Dribbel-Verhalten teilweise bei Annahme und Abgabe des Balles von zwei Pässen erkannt. Bei 5 Takten wurden die kurzen Dribbel-Manöver nicht mehr erkannt.

Gründe, z.B. den Grund, gegnerischen Spielern auszuweichen. Eine wesentliche Änderung der Laufrichtung des Dribblers wird als Beendigung seines aktuellen Dribbel-Verhaltens angesehen. Dies führt zu der Bedingung, daß sich ein Spieler während eines Dribbel-Manövers relativ **geradlinig bewegen** muß. Diese Bedingung wird mit

$$\text{straightMovement}\,(p_1, t_0, t_n, playerMovement)$$

bezeichnet, wobei $playerMovement \in \mathbb{V}$ der resultierende Bewegungsvektor vom Spieler p_1 im Zeitintervall t_0 bis t_n ist.

Wie beim Paß-Verhalten wird auch das Dribbel-Verhalten durch eine Änderung des Spielzustandes abgebrochen. Die einzige Ausnahme ist der Start eines Dribbel-Manövers bei einem Freistoß etc.. Es muß also **außer im ersten Takt** immer der **Spielzustand *play on*** gelten:

$$\text{virtuallyPlayOn}\,(t_0, t_n).$$

Damit ergibt sich die Definition für die Verhaltens-Ausprägung Dribbeln mit ständiger Ballkontrolle (Verhaltens-Muster 3).

Verhaltens-Muster 3: Dribbeln von Spieler p_1 mit ständiger Ballkontrolle

$$\begin{aligned}\text{ballControlDribble}\,(p_1, t_0, t_n, playerMovement) :- \ & \text{XballControl}\,(p_1, t_0, t_n),\\ & \text{straightMovement}\,(p_1, t_0, t_n, playerMovement),\\ & \text{virtuallyPlayOn}\,(t_0, t_n),\ \text{minTime}_4\,(t_0, t_n).\end{aligned}$$

4.4.1.2 Dribbeln mit vorübergehender Aufgabe der Ballkontrolle

Bei dieser Ausprägung des Passes muß die ständige Kontrolle des Balles durch den Spieler nicht mehr gegeben sein. Der Spieler muß die **Kontrolle des Balles** nur **bei Start und bei Ende** dieser Dribbel-Ausprägung haben. Dies wird durch die beiden Bedingungen

$$\text{XballControl}\,(p_1, t_0, t_1), \text{ und}$$

$$\text{XBallControl}_1\,(p_1, t_n, t_n)$$

gewährleistet, wobei t_0 den Start- und t_n den Endzeitpunkt des Verhaltens kennzeichnet.

In der Zwischenzeit darf natürlich **kein anderer Spieler** den **Ball kontrollieren**, eine Kontrolle des Balles durch den Spieler p_1 ist aber erlaubt. Diese Bedingung wird durch den Ausdruck

$$\text{XnearBall}\,(p_1, t_2, t_3)$$

dargestellt, der eine oder-Verknüpfung der beiden Ausdrücke XballControl und ballFree ist.

Beim Paß wurde die Bedingung aufgenommen, daß sich der Ball nach Verlassen der Kontrolle durch den Kicker schnell von diesem entfernt (Bedingung ballFastDeparting). Beim Dribbeln mit vorübergehender Aufgabe der Ballkontrolle entfernt sich der Ball dagegen langsam vom Dribbler. Langsam bedeutet dabei mit einer Geschwindigkeit, so daß die Bedingung ballFastDeparting nicht gilt. Beim Dribbeln kann sich der Ball mehrmals vom Spieler entfernen, solange

er zwischendurch nur von keinem anderen Spieler kontrolliert wird. Demnach muß für das gesamte Zeitintervall (t_2, t_3) sichergestellt werden, daß sich der **Ball nur langsam vom Spieler p_1 entfernt**. Diese Bedingung bezeichen wir mit

$$\text{ballSlowDeparting}\,(p_1, t_2, t_3).$$

Die anderen Bedingungen ändern sich gegenüber der Dribbel-Ausprägung mit ständiger Ballkontrolle nicht, so daß sich die im Verhaltens-Muster 4 dargestellte Definition für die Dribbel-Ausprägung mit vorübergehender Aufgabe der Ballkontrolle ergibt.

Verhaltens-Muster 4: Dribbeln von Spieler p_1 mit vorübergehender Aufgabe der Ballkontrolle

$$\begin{aligned}\text{dribble}\,(p_1, t_0, t_n, playerMovement) :- \ & \text{XballControl}\,(p_1, t_0, t_1),\\ & \text{XnearBall}\,(p_1, t_2, t_3),\ \text{XBallControl}_1\,(p_1, t_n, t_n),\ \text{follow}\,(t_1, t_2),\\ & \text{follow}\,(t_3, t_n),\ \text{ballSlowDeparting}\,(p_1, t_2, t_3),\\ & \text{straightMovement}\,(p_1, t_0, t_n, playerMovement),\\ & \text{virtuallyPlayOn}\,(t_0, t_n),\ \text{minTime}_4\,(t_0, t_n).\end{aligned}$$

Ein interessantes Attribut des Dribbel-Manövers ist die Durchschnittsgeschwindigkeit des Spielers, welche durch die Dribbel-Dauer $(t_n - t_0)$ und durch die Gesamtbewegung des Spielers $(playerMovement.length)$ gegeben ist.

4.4.1.3 Weitere Ausprägungen des Dribbel-Verhaltens

Mit dem Attribut $playerMovement$ kann eine weitere Ausprägung des Dribbel-Verhaltens spezifiziert werden.

Ball halten: Bei dieser Dribbel-Ausprägung bewegt sich der Spieler gar nicht oder nur wenig mit dem Ball vorwärts, es gilt $playerMovement.length \approx 0$. Ein solches Dribbel-Verhalten kann in Situationen sinnvoll sein, in denen der Dribbler bereits eine sehr gute Position erreicht hat und auf eine bessere Positionierung der anderen Spieler seines Teams wartet.

Das Verhaltens-Muster Dribbeln kann auch in den in Abschnitt 4.4.4 auf Seite 72 beschriebenen Start- und Fehler-Ausprägungen vorkommen.

4.4.2 Das Verhaltens-Muster Torschuß

Die Vorraussetzung für einen Schuß auf das Tor ist die **alleinige Kontrolle über** den **Ball** durch einen Spieler p_1. Zur Vorbereitung eines Torschußes sind 2 Takte Kontrolle über den Ball ausreichend, Bedingung

$$\text{XBallControl}_2\,(p_1, t_0, t_1).$$

Nach dieser Ballkontrolle durch den Kicker darf der **Ball durch keinen Spieler kontrolliert** werden, Bedingung

$$\text{ballFree}\,(t_2, t_3).$$

Die meisten Torschüsse haben eine hohe Geschwindigkeit, aber es gibt auch welche, bei denen der Spieler bereits den Torwart umspielt hat und den Ball mit

geringer Geschwindigkeit ins Tor befördert. Die **anfängliche Ballgeschwindigkeit** beschreibt dabei den Torschuß genauer und wird mit dem Ausdruck

$$\text{ballDeparting}\,(p_1, t_2, ballSpeed)$$

festgehalten, wobei $ballSpeed \in \mathbb{V}$ die Geschwindigkeit des Balles ist, nachdem dieser im Zeittakt t_2 die Kontrolle des Spielers p_1 verlassen hat.

Desweiteren muß sich der Ball auch in die Richtung des gegnerischen Tores bewegen, damit überhaupt eine Chance auf ein Tor besteht. Das **Tor** muß sich also **in** dem **Bereich** befinden, in den der **Ball gekickt** wurde[15]. Dies wird mit der Bedingung

$$\text{goalInKickRegion}\,(p_1, t_2, shotTarget)$$

ausgedrückt, wobei t_2 der erste Zeitpunkt ist, zu dem der Kicker den Ball nicht mehr kontrolliert und $shotTarget \in \mathbb{V}$ die Position[16] auf dem Spielfeld bezeichnet, an der der Ball vorraussichtlich die Torlinie überqueren wird.

Schließlich muß der **Ball ins Tor treffen**, was durch die Bedingung

$$\text{goal}\,(p_1, t_n)$$

ausgedrückt wird, wobei p_1 den Torschützen bezeichnet und t_n der Zeitpunkt des Tortreffers ist. Beim Treffen des Tores wechselt der Spielzustand im *SoccerServer* in den Spielzustand *goal*, um den Treffer anzuzeigen. Demnach stellt die Bedingung $\text{goal}\,(p_1, t_n)$ auch einen anderen Spielzustand als *play on* dar.

Von Beginn des Verhaltens bis zum Zeitpunkt, an dem das Tor erzielt wurde, darf der **Spielzustand** sich **außer beim Start nicht ändern**, Bedingung

$$\text{virtuallyPlayOn}\,(t_0, t_3).$$

Zusammen mit den entsprechenden follow-Bedingungen ergibt sich die Definition für das Verhaltens-Muster Torschuß (Verhaltens-Muster 5).

Verhaltens-Muster 5: Torschuß von Spieler p_1

$$\begin{aligned}\text{scoreGoal}\,(p_1, t_0, t_n, ballSpeed, shotTarget) :- \;& \text{XBallControl}_2\,(p_1, t_0, t_1),\\ & \text{ballFree}\,(t_2, t_3),\ \text{goal}\,(p_1, t_n),\ \text{follow}\,(t_1, t_2),\ \text{follow}\,(t_3, t_n),\\ & \text{ballDeparting}\,(p_1, t_2, ballSpeed),\\ & \text{goalInKickRegion}\,(p_1, t_2, shotTarget),\ \text{virtuallyPlayOn}\,(t_0, t_3).\end{aligned}$$

Ausprägungen des Torschuß-Verhaltens können z.B. über die beiden Attribute *ballSpeed* und *shotTarget* festgelegt werden.

4.4.3 Das Verhaltens-Muster Klären

Das Verhaltens-Muster Klären steht für eine weiträumige Beförderung des Balles durch einen Spieler p_1 in den freien Raum mit der Erwartung, daß ein Spieler p_2 der eigenen Mannschaft als erster wieder die Kontrolle über den Ball erlangt. Dieser Spieler kann dabei auch der Kicker selbst sein.

[15] Wenn der Ball nicht vorher abgefangen wird, erreicht er mit ausreichend großer Geschwindigkeit das gegnerische Tor oder das gegnerische Toraus in unmittelbarer Nähe des Tores.

[16] Mit dem Vektor *shotTarget* wird das genaue Ziel des Schußes für die Verhaltens-Vorhersage festgehalten.

Damit muß gelten, daß der Spieler p_1 den **Ball** am Anfang **allein kontrolliert**, Bedingung

$$\text{XBallControl}_2\,(p_1, t_0, t_1),$$

der **Ball** danach für einen Zeitraum **von keinem Spieler kontrolliert** wird, Bedingung

$$\text{ballFree}\,(t_2, t_3),$$

bis später wieder ein Spieler die **alleinige Kontrolle über** den **Ball** erlangt, Bedingung

$$\text{XBallControl}_1\,(p_2, t_n, t_n).$$

Die **Spieler** p_1 und p_2 müssen dabei **zum selben Team** gehören, Bedingung

$$\text{sameTeam}\,(p_1, p_2)$$

und das Verhalten darf **nicht** vorzeitig **durch** einen **Spielzustandswechsel abgebrochen** werden, Bedingung

$$\text{virtuallyPlayOn}\,(t_0, t_n).$$

Die geforderte weiträumige Beförderung des Balles wird durch eine **hohe Anfangsgeschwindigkeit des Balles** erreicht, Bedingung

$$\text{ballFastDeparting}\,(p_1, t_2, \mathit{ballSpeed}).$$

Dieses Verhaltens-Muster unterscheidet sich von dem Torschuß- und dem Paß-Verhalten vor allem dadurch, daß der Ball in den freien Raum gekickt wird, sich also **kein Spieler der eigenen Mannschaft oder das gegnerische Tor im Kickbereich** befinden. Wir bezeichnen diese Bedingung mit dem Ausdruck

$$\text{nothingInKickRegion}\,(p_1, t_2).$$

Es ergibt sich die in Verhaltens-Muster 6 dargestellte Definition.

Verhaltens-Muster 6: Klären von Spieler p_1

$$\begin{aligned}\text{clear}\,(p_1, t_0, t_n, \mathit{ballSpeed}) :-\ & \text{XBallControl}_2\,(p_1, t_0, t_1),\ \text{ballFree}\,(t_2, t_3),\\ & \text{XBallControl}_1\,(p_2, t_n, t_n),\ \text{follow}\,(t_1, t_2),\ \text{follow}\,(t_3, t_n),\\ & \text{ballFastDeparting}\,(p_1, t_2, \mathit{ballSpeed}),\ \text{nothingInKickRegion}\,(p_1, t_2),\\ & \text{sameTeam}\,(p_1, p_2),\ \text{virtuallyPlayOn}\,(t_0, t_n).\end{aligned}$$

Das Klären-Verhalten kann dahingehend unterschieden werden, ob der Spieler, der die Kontrolle über den Ball erlangt der Kicker selbst, oder ein anderer Spieler ist. Damit ergeben sich zwei Ausprägungen dieses Verhaltens-Musters.

Vorlegen: Der Kicker erlangt selbst die Kontrolle über den Ball, er hat sich also den Ball vorgelegt. Diese Ausprägung ist durch eine Aufnahme der Bedingung $\text{samePlayer}\,(p_1, p_2)$ gegeben.

Mitspieler: Ein Mitspieler des Kickers erlangt die Kontrolle über den Ball, zusätzlich gilt also die Bedingung $\text{notSame}\,(p_1, p_2)$.

4.4.4 Gleiche Ausprägungen verschiedener Verhaltens-Muster

In diesem Abschnitt werden die Ausprägungen der Verhaltens-Muster Paß, Dribbeln, Torschuß und Klären vorgestellt, die für alle diese Verhaltens-Muster gleich sind. Dies sind Start-Ausprägungen, mit denen die Start-Vorraussetzungen von Verhalten unterschieden werden und Fehler-Ausprägungen, mit denen erfolgloses Verhalten genauer unterschieden wird.

4.4.4.1 Verschiedene Start-Ausprägungen

Die Verhalten Paß, Dribbeln, Torschuß und Klären finden meist direkt aus dem Spiel heraus statt, d.h. sie starten im Spielzustand *play on*. Sie können aber auch direkt aus einem Freistoß etc. entstehen, also in einem anderen Spielzustand als *play on* starten. Es können zwei Ausprägungen von Verhalten bzgl. deren Start-Voraussetzungen eingeführt werden.

Im Spiel: Der Start des Verhaltens findet im Spielzustand *play on* statt. Wir bezeichnen diese Einschränkung für den Zeitpunkt t_0 mit dem Ausdruck

$$\text{playOn}\,(t_0).$$

Dieser Ausdruck wird für diese Ausprägung als zusätzliche Bedingung aufgenommen.

Freistoß etc.: Der Start des Verhaltens findet nicht im Spielzustand *play on* statt. Mit der zusätzlichen Bedingung

$$\neg\,\text{playOn}\,(t_0)$$

werden alle diese Ausprägungen beschrieben.

Diese Unterteilung der Verhalten ist sinnvoll, da bei Freistössen etc. im Gegensatz zur Ausprägung „Im Spiel“ dem Team ausreichend Zeit zur Verfügung steht, um sich beliebig zu positionieren und damit dem IP prinzipiell andere Ausgangssituationen für die Verhaltensauswahl zur Verfügung stehen.

4.4.4.2 Verschiedene Fehler-Ausprägungen

Es soll nicht nur erfolgreiches Verhalten, sondern auch erfolgloses Verhalten erkannt werden. Desweiteren soll bei erfolglosen Verhalten auch der Grund des Mißerfolges ermittelt werden.

Bei den Verhaltens-Mustern Paß, Dribbeln, Torschuß und Klären ist ein erfolgloses Verhalten durch den Kontrollverlust des Balles an das gegnerische Team gegeben. Dies ist der Fall, wenn

(a) ein gegnerischer Spieler den Ball kontrolliert bzw.

(b) der Spielzustand so wechselt, daß ein gegnerischer Spieler demnächst den Ball kontrollieren wird (z.B. durch einen Schuß des Balles ins Aus).

Beide Fälle werden jeweils als gesonderte Verhaltens-Ausprägung für die vier Verhaltens-Muster Paß, Dribbeln, Torschuß und Klären eingeführt.

Kontrolle des Balles durch gegnerischen Spieler (a) In diesem Fall wird das „Ziel“ des Verhaltens nicht erreicht, weil ein gegnerischer Spieler die Kontrolle über den Ball erlangt. Das „Ziel“ eines Verhaltens-Musters ist dabei durch eine Teilmenge der Bedingungen des Verhaltens-Musters gegeben. Die „Ziel“-Bedingungen für die vier Verhaltens-Muster sind:

- Paß: $\text{XBallControl}_1\,(p_2, t_n, t_n)$, $\text{sameTeam}\,(p_1, p_2)$, $\text{notSame}\,(p_1, p_2)$
- Dribbeln[17]: $\text{XBallControl}_1\,(p_1, t_n, t_n)$, $\text{minTime}_4\,(t_0, t_n)$
- Torschuß: $\text{goal}\,(p_1, t_n)$
- Klären: $\text{XBallControl}_1\,(p_2, t_n, t_n)$, $\text{sameTeam}\,(p_1, p_2)$

Zur Darstellung der Kontrolle des Balles durch einen gegnerischen Spieler werden die „Ziel“-Bedingungen der jeweiligen Verhaltens-Muster durch die beiden Bedingungen $\text{ballControl}_1\,(p_3, t_n, t_n)$ und $\neg\,\text{sameTeam}\,(p_1, p_3)$ ersetzt. Mit der Bedingung

$$\text{ballControl}_1\,(p_3, t_n, t_n)$$

wird ausgesagt, daß der Spieler p_3 die **Kontrolle über** den **Ball** im Zeittakt t_n hat, aber nicht der einzige Spieler sein muß, für den dies gilt.

Für den Torschuß ist diese Fehler-Ausprägung (gegnerischer Spieler kontrolliert Ball) beispielhaft mit dem Verhaltens-Muster 7 dargestellt. Zur Verdeutlichung der Veränderungen zum Verhaltens-Muster Torschuß sind entfernte Ereignisse durchgestrichen und aufgenommene Ereignisse unterstrichen dargestellt.

Verhaltens-Muster 7: Erfolgloser Torschuß von Spieler p_1, abgefangen durch Spieler p_3

$\text{scoreGoal}\,(p_1, t_0, t_n, \mathit{ballSpeed}, \mathit{shotTarget}) :-$ $\text{XBallControl}_2\,(p_1, t_0, t_1)$,
$\text{ballFree}\,(t_2, t_3)$, ~~$\text{goal}\,(p_1, t_n)$~~, $\text{follow}\,(t_1, t_2)$, $\text{follow}\,(t_3, t_n)$,
$\text{ballDeparting}\,(p_1, t_2, \mathit{ballSpeed})$,
$\text{goalInKickRegion}\,(p_1, t_2, \mathit{shotTarget})$, $\text{virtuallyPlayOn}\,(t_0, t_3)$,
$\underline{\text{ballControl}_1\,(p_3, t_n, t_n)}$, $\underline{\neg\,\text{sameTeam}\,(p_1, p_3)}$.

Wechsel des Spielzustandes (b) Bei dieser Fehler-Ausprägung ist das Verhalten erfolglos, weil sich der Spielzustand verändert hat, bevor das „Ziel“ des Verhaltens-Musters erreicht wurde. Um diese Ausprägung eines Verhaltens-Musters darzustellen, wird dessen Zielbedingung gestrichen und die Bedingung $\text{virtuallyPlayOn}\,(t_0, t_n)$ durch die Bedingungen $\text{virtuallyPlayOn}\,(t_0, t_n - 1)$ und $(\neg\,\text{playOn}\,(t_n) \wedge \neg\,\text{goal}\,(p_1, t_n))$ ersetzt. Beim Verhaltens-Muster Torschuß wird nur die Zielbedingung durch die Bedingung $(\neg\,\text{playOn}\,(t_n) \wedge \neg\,\text{goal}\,(p_1, t_n))$ ersetzt.

[17] Bei dem Dribbel-Verhalten mit ständiger Ballkontrolle ist die gleiche „Ziel“-Bedingung gegeben, nachdem die Dribbel-Bedingung $\text{XballControl}\,(p_1, t_0, t_n)$ in die 3 Bedingungen $\text{XballControl}\,(p_1, t_0, t_1)$, $\text{XBallControl}_1\,(p_1, t_n, t_n)$, und $\text{follow}\,(t_1, t_n)$ zerlegt wurde.

Diese Verhaltens-Ausprägung kann dabei noch weiter unterteilt werden, indem die neue Bedingung $(\neg\,\mathrm{playOn}\,(t_n) \wedge \neg\,\mathrm{goal}\,(p_1, t_n))$ durch eine andere, den Spielzustand genauer spezifizierendere Bedingung, ausgetauscht wird. Einige solcher Ausprägungen sind:

Aus: Das Verlassen des Spielfeldes durch den Ball im Zeittakt t_n wird durch die Bedingung $\mathrm{kickIn}\,(t_n)$ beschrieben.

Abseits: Der Wechsel des Spielzustandes im Zeittakt t_n aufgrund von Abseits wird durch die Bedingung $\mathrm{offside}\,(t_n)$ gekennzeichnet.

Gehalten: Das Fangen des Balles durch den gegnerischen Torwart im Zeittakt t_n wird durch die Bedingung $\mathrm{catch}\,(t_n)$ dargestellt.

Das Auftreten fast aller Fehler-Ausprägungen im Spiel ist dabei auch von der Spielstärke des gegnerischen Teams abhängig, einzige Ausnahme ist die Fehler-Ausprägung „Aus“.

4.4.5 Zusammenfassung der bisherigen Verhaltens-Muster

Bei allen bisher eingeführten Verhaltens-Mustern kontrolliert an deren Start und Ende jeweils genau ein Spieler den Ball oder der Ball landet im Tor. In diesem Abschnitt wird untersucht, ob durch die vier Verhaltens-Muster Paß, Dribbeln, Torschuß und Klären die Möglichkeiten des Transfers eines Balles durch einen Spieler vollständig abgebildet sind. Desweiteren wird ergründet, ob die Definitionen der Verhaltens-Muster eindeutig sind. Dazu werden zunächst die Gemeinsamkeiten und Unterschiede der vier Verhaltens-Muster betrachtet.

Bei diesen Verhaltens-Mustern wird ein Transfer des Balles von einem Kicker vorgenommen. Die Verhaltens-Muster können anhand der folgenden Attribute unterschieden werden.

Empfänger: Der Empfänger des Balles kann der Spieler selbst, ein Mitspieler, das gegnerische Tor oder im Fehlerfall sogar ein Gegner sein. Dies wurde mit Hilfe der Ausdrücke samePlayer, sameTeam, notSame und goal definiert.

Geschwindigkeit des Balles: Die Ballgeschwindigkeit bei der anfänglichen Entfernung des Balles vom Kicker kann schnell, langsam oder beliebig sein. Die unterschiedlichen Ballgeschwindigkeiten wurden durch die Bedingungen ballFastDeparting, ballSlowDeparting, und ballDeparting gekennzeichnet. Dabei schließen sich die Bedingungen ballFastDeparting und ballSlowDeparting gegenseitig aus.

Objekte im Kickbereich: In dem Bereich, in den der Ball gekickt wird, können sich Mitspieler, das gegnerische Tor, oder keines von beiden befinden. Die Einschränkungen bzgl. der erlaubten Objekte im Kickbereich wurden mit den drei Ausdrücken teammateInKickRegion, goalInKickRegion und nothingInKickRegion vorgenommen. Während der letzte Ausdruck – wenn er gilt – die ersten beiden Ausdrücke ausschließt und umgekehrt,

können die beiden ersten Ausdrücke auch zur gleichen Zeit gelten. Der Kickbereich ist dabei von der Geschwindigkeit des Balles abhängig. Ein Spieler befindet sich im Kickbereich, wenn er sich in dem Gebiet aufhält, welches ihm die Möglichkeit gibt, den Ball relativ frühzeitig abzufangen. Das gegnerische Tor befindet sich im Kickbereich, wenn der Ball mit einer nicht zu geringen Geschwindigkeit das Tor treffen würde, wenn er nicht vorher abgefangen wird. Die Werte zur Bestimmung des Kickbereiches wurden empirisch ermittelt.

Tabelle 4.1 fasst die Unterschiede der vier Verhaltens-Muster bzgl. der Attribute Empfänger, Geschwindigkeit des Balles und „Objekte im Kickbereich“ zusammen.

	Empfänger	Geschwindigkeit des Balles	Objekte im Kickbereich
Paß	Mitspieler	schnell	mindestens Mitspieler
Dribbeln	Spieler selbst	langsam	beliebig
Torschuß	Gegner-Tor	beliebig	mindestens Gegner-Tor
Klären	Mitspieler oder Spieler selbst	schnell	kein Tor und kein Mitspieler

Tabelle 4.1: Zusammenfassung der vier Balltransfers

In der Tabelle 4.2 auf der nächsten Seite sind alle Kombinationen bzgl. dieser drei Attribute aufgeführt. Dabei bedeuten „Mitspieler“ bzw. „Gegner-Tor“, daß sich nur Mitspieler bzw. nur das gegnerische Tor im Kickbereich befinden. Die Fälle in denen sich Mitspieler und Tor im Kickbreich befinden, wurden weggelassen. Das Wort „nichts“ in der Spalte „Objekte im Kickbereich“ bedeutet, daß sich weder Gegner-Tor noch Mitspieler im Kickbereich befinden. In der letzten Spalte Verhalten ist angegeben, um welches Verhalten es sich jeweils handelt. Mit dem Zeichen „—“ wird dargestellt, daß für diese Bedingungen kein Verhaltens-Muster spezifiziert wurde, diese Problemfälle werden im Anschluß diskutiert. Das Wort „unmöglich“ sagt aus, daß es sich bei den Bedingungen um eine unmögliche Kombination handelt.

Mit der Tabelle ist gezeigt, daß die Zuordnung zu den vier Verhaltens-Mustern Paß, Dribbeln, Torschuß und Klären im **Erfolgsfall** eindeutig ist. Selbst wenn sich Mitspieler und Tor gleichzeitig im Kickbereich befinden, ergibt sich keine Mehrdeutigkeit der Verhalten. Leider ist die Zuordnung zu den vier Verhaltens-Mustern nicht vollständig. Es gibt Kombinationen bzgl. der drei Attribute die nicht durch eines der vier Verhaltens-Muster beschrieben sind. Diese 6 Kombinationen werden im folgenden genauer betrachtet:

Problem 1: Bei diesem Problem stellt sich die Frage, warum nicht der Mitspieler als Empfänger aufgetreten ist, er befand sich ja im Kickbereich. Eine naheliegende Erklärung ist, daß der Mitspieler sich nur zufällig im Kickbereich befand und der Kicker geplant hatte ein Klären-Manöver durch-

Empfänger	Geschwindigkeit des Balles	Objekte im Kickbereich	Verhalten
Spieler selbst	langsam	Mitspieler	Dribbeln
Spieler selbst	langsam	Gegner-Tor	Dribbeln
Spieler selbst	langsam	nichts	Dribbeln
Spieler selbst	schnell	Mitspieler	— (Problem 1)
Spieler selbst	schnell	Gegner-Tor	— (Problem 2)
Spieler selbst	schnell	nichts	Klären
Mitspieler	langsam	Mitspieler	— (Problem 3)
Mitspieler	langsam	Gegner-Tor	— (Problem 4)
Mitspieler	langsam	nichts	— (Problem 5)
Mitspieler	schnell	Mitspieler	Paß
Mitspieler	schnell	Gegner-Tor	— (Problem 6)
Mitspieler	schnell	nichts	Klären
Tor	langsam	Mitspieler	unmöglich[18]
Tor	langsam	Gegner-Tor	Torschuß
Tor	langsam	nichts	unmöglich[18]
Tor	schnell	Mitspieler	unmöglich[18]
Tor	schnell	Gegner-Tor	Torschuß
Tor	schnell	nichts	unmöglich[18]

Tabelle 4.2: Untersuchung auf Vollständigkeit und Eindeutigkeit

zuführen. Es ist somit möglich dieses Verhalten als **Klären** (Ausprägung Vorlegen) zu bezeichnen.

Problem 2: Analog zu Problem 1 taucht hier die Frage auf, warum der Ball nicht ins Tor gegangen ist, sondern der Kicker den Ball abgefangen hat. Eine sinnvolle Erklärung ist, daß der Spieler vor dem Torschuß den Ball näher an das Tor heranbringen wollte. Dieses Verhalten könnte demnach auch als **Klären** (Ausprägung Vorlegen) bezeichnet werden.

Problem 3: Abgesehen von der Ballgeschwindigkeit handelt es sich hier um einen **Paß**. Eine eindeutige Erklärung für die geringe Ballgeschwindigkeit kann nicht gegeben werden. Einem Beobachter ist es nur schwer möglich zu entscheiden, ob die geringe Ballgeschwindigkeit beabsichtigt war oder auf ein Versehen zurückzuführen ist.

Problem 4: Bei dieser Kombination sollte der Spieler selbst oder das Gegner-Tor als Empfänger auftreten und nicht ein Mitspieler. Eine mögliche Erklärung für solch ein Verhalten ist, daß der ballführende Spieler von einem Mannschaftskameraden abgelöst wird (wie beim Staffellauf), weil seine Kraftreserven erschöpft sind. Dieses Verhalten könnte man folglich als **Klären** (Ausprägung Mitspieler) bezeichnen.

[18] Wenn sich das Tor nicht im Kickbereich befindet, dann kann der Ball unmöglich ins Tor gelangen. Bei der Bestimmung des Kickbereichs wurden zufällige Abweichungen der Ballbewegung bereits berücksichtigt.

Problem 5: Dieses Problem gleicht dem Problem 4, bis auf die Objekte die sich im Kickbereich befinden. Die Erklärung von Problem 4 läßt sich auf dieses Problem anwenden, so daß wir diese Kombination als **Klären** (Ausprägung Mitspieler) bezeichnen könnten.

Problem 6: Bei dieser Kombination stellt sich die Frage, wie es ein Mitspieler, der sich nicht im Kickbereich aufhält schafft, den sich schnell entfernenden Ball abzufangen bevor dieser das Tor erreicht. Es ist denkbar, daß sich das Tor nur zufällig im Kickbereich befunden hat und das Verhalten als ein Klären-Manöver geplant war.

Demnach ist es vorstellbar dieses Verhalten als **Klären** (Ausprägung Mitspieler) zu bezeichnen. In Situationen in denen sich Mitspieler und das gegnerische Tor im Kickbereich befinden, geht damit aber die Eindeutigkeit verloren. Betrachtet man das Tor im Kickbereich wird ein Klären-Verhalten, betrachtet man den Mitspieler im Kickbereich, ein Paß-Verhalten erkannt.

Bei diesen 6 Problemfällen wurde immer mit angegeben, wie ein solches Verhalten bezeichnet werden könnte. Es stellt sich die Frage, ob dies sinnvolle Bezeichnungen sind. Desweiteren haben wir festgestellt, daß die Eindeutigkeit der Verhalten nicht gesichert werden kann, wenn die Problemfälle den bisherigen Verhaltens-Mustern zugeordnet werden.

Es werden deshalb alle Problemfälle (Balltransfers die nicht durch die vier Verhaltens-Muster Paß, Dribbeln, Torschuß und Klären beschrieben werden können) in einem gesonderten Verhaltens-Muster aufgezeichnet. Damit können die Ursachen für die Problemfälle und ihre Häufigkeit bestimmt werden.

Folglich ist die Untergliederung der Balltransfers in Pässe, Dribbeln, Torschuß und Klären nicht vollständig, aber im Erfolgsfall eindeutig.

Im **Fehlerfall** sieht dies anders aus. Tabelle 4.3 zeigt alle Kombinationen bzgl. der Attribute Geschwindigkeit des Balles und „Objekte im Kickbereich“ für den Fall, daß der Transfer fehlerhaft (Kontrolle des Balles durch gegnerischen Spieler oder Wechsel des Spielzustandes) war. Der Fehlerfall ist in der Tabelle durch einen Gegner als Empfänger dargestellt. In der letzten Spalte ist angegeben, welchem Verhalten der Fehlschlag zuzuordnen ist.

Empfänger	Geschwindigkeit des Balles	Objekte im Kickbereich	Verhalten
Gegner	langsam	Mitspieler	Dribbeln
Gegner	langsam	Gegner-Tor	Dribbeln oder Torschuß
Gegner	langsam	nichts	Dribbeln
Gegner	schnell	Mitspieler	Paß
Gegner	schnell	Gegner-Tor	Torschuß
Gegner	schnell	nichts	Klären

Tabelle 4.3: Fehlerfälle bei Ballverlust

Die Zuordnung zu den Verhalten ist demnach im Fehlerfall vollständig. Leider ist diese Zuordnung nicht eindeutig. So kann ein fehlerhafter langsamer Transfer mit dem Tor im Kickbereich entweder durch ein Dribbeln oder durch einen Torschuß entstanden sein. Bei einem fehlerhaften schnellen Transfer mit Tor und Mitspieler im Kickbereich kann es sich um einen Torschuß oder um einen Paß gehandelt haben. Eine eindeutige Klassifizierung solcher Fehler wäre vorteilhaft, dies ist aber selbst für einen Menschen bei einem Fußballspiel sehr schwierig und wird daher nicht weiter verfolgt.

Die Unterteilung der Balltransfers in die vier Verhaltens-Muster Paß, Dribbeln, Torschuß und Klären führt damit im Erfolgsfall zu einer nicht vollständigen aber eindeutigen und im Fehlerfall zu einer vollständigen aber nicht eindeutigen Zerlegung.

4.4.6 Das Verhaltens-Muster Doppelpaß

Ein Doppelpaß beschreibt ein Zusammenspiel zweier Spieler des selben Teams mit dem Ziel, die gegnerische Verteidigung auszumanövrieren. Dabei passt der den Ball kontrollierende Spieler p_1 den Ball zu einem Spieler p_2, der dann den Ball sofort zum Spieler p_1 zurückpasst[19].

Ein Doppelpaß ist damit aus 2 Pässen zusammengesetzt und könnte daher rekursiv mit Hilfe des Verhaltens-Musters Paß definiert werden. Das Verhaltens-Muster Doppelpaß würde dann aus 2 Paß-Verhaltens-Mustern, einem zeitlichen und einem weiteren Ereignis-Muster bestehen. Da der Formalismus und die im Abschnitt 4.5 beschriebene Realisierung aber davon ausgehen, daß sich ein Verhaltens-Muster nur aus Ereignis-Mustern zusammensetzt, wird der Doppelpaß u.a. mit schon bekannten Bedingungen vom Paß definiert, die hier nicht wiederholt werden.

Bei der Abfolge von zwei Pässen sollte das Ende vom ersten Paß mit dem Beginn des zweiten Passes in einem Ausdruck zusammengefasst werden. Die Ballkontrolle des Spielers, der zwischenzeitlich den Ball kontrolliert (Spieler p_2), wird daher auf 3 Takte beschränkt, Bedingung

$$\text{XBallControl}_3\,(p_2, t_4, t_5).$$

Dies entspricht genau der Summe der Zeitbeschränkungen der Annahme eines Passes und der Vorbereitung eines Passes.

Auch wenn sich der Doppelpaß aus zwei Pässen zusammensetzt und damit größtenteils aus Bedingungen besteht die schon vom Verhaltens-Muster Paß bekannt sind, gibt es eine neue Bedingung für den Doppelpaß. Beim Paß wurde nach dem Kicken des Balles der Kicker nicht mehr betrachtet. Beim Doppelpaß ist jedoch der Spieler p_1 weiterhin interessant, da er den Ball wieder kontrollieren wird und damit weiter am Verhalten teilnimmt. Der **Spieler** p_1 fängt gleich nach dem Kicken damit an, sich zu **positionieren**. Dies wird mit dem Ausdruck

$$\text{positioning}\,(p_1, t_2, t_5, positioningMovement),$$

bezeichnet, wobei $positioningMovement \in \mathbb{V}$ der resultierende Bewegungsvek-

[19] Ein zwischenzeitliches Dribbeln durch den Spieler p_2 wird dabei nicht erlaubt.

tor des Spielers p_1 im Zeitintervall von t_2 bis t_5 ist.

Weitere neue Bedingungen sind für das Verhaltens-Muster Doppelpaß nicht nötig, so daß sich die im Verhaltens-Muster 8 dargestellte Definition ergibt. Dabei bezeichnen $bS1$ und $bS2$ Ballgeschwindigkeiten ($ballSpeed$) und $pM1$ und $pM2$ resultierende Bewegungsvektoren der Spieler ($playerMovement$).

Verhaltens-Muster 8: Doppelpaß zwischen Spieler p_1 und Spieler p_2

$$\begin{aligned}\text{oneTwo}\,(p_1, p_2, t_0, t_n, bS1, pM1, positioningMovement, bS2, pM2) :- \\ &\text{XBallControl}_2\,(p_1, t_0, t_1),\ \text{ballFree}\,(t_2, t_3),\ \text{XBallControl}_3\,(p_2, t_4, t_5),\\ &\text{ballFree}\,(t_6, t_7),\ \text{XBallControl}_1\,(p_1, t_n, t_n),\ \text{follow}\,(t_1, t_2),\\ &\text{follow}\,(t_3, t_4),\ \text{follow}\,(t_5, t_6),\ \text{follow}\,(t_7, t_n),\\ &\text{ballFastDeparting}\,(p_1, t_2, bS1),\ \text{teammateInKickRegion}\,(p_1, t_2),\\ &\text{movement}\,(p_2, t_2, t_4, pM1),\\ &\text{positioning}\,(p_1, t_2, t_5, positioningMovement),\\ &\text{ballFastDeparting}\,(p_2, t_6, bS2),\ \text{teammateInKickRegion}\,(p_2, t_6),\\ &\text{movement}\,(p_1, t_6, t_n, pM2),\ \text{sameTeam}\,(p_1, p_2),\ \text{notSame}\,(p_1, p_2),\\ &\text{virtuallyPlayOn}\,(t_0, t_n).\end{aligned}$$

4.4.7 Das Verhaltens-Muster Abseitsfalle

Eine Abseitsfalle hat den Zweck, den gegnerischen Angriff zu verzögern, indem einer oder mehrere gegnerische Spieler in eine Abseitsposition gebracht werden. Um dies zu erreichen müssen die Verteidiger des Teams, die die Abseitsfalle stellen, koordiniert agieren. Unkoordinierte Aktionen einzelner Verteidiger sind selten von Erfolg gekrönt.

Ein gegnerischer Spieler befindet sich im Abseits, wenn er sich zwischen der Abseitslinie des verteidigenden Teams und der Torlinie des verteidigenden Teams befindet. Die Abseitslinie ist durch 3 Bedingungen bestimmt:

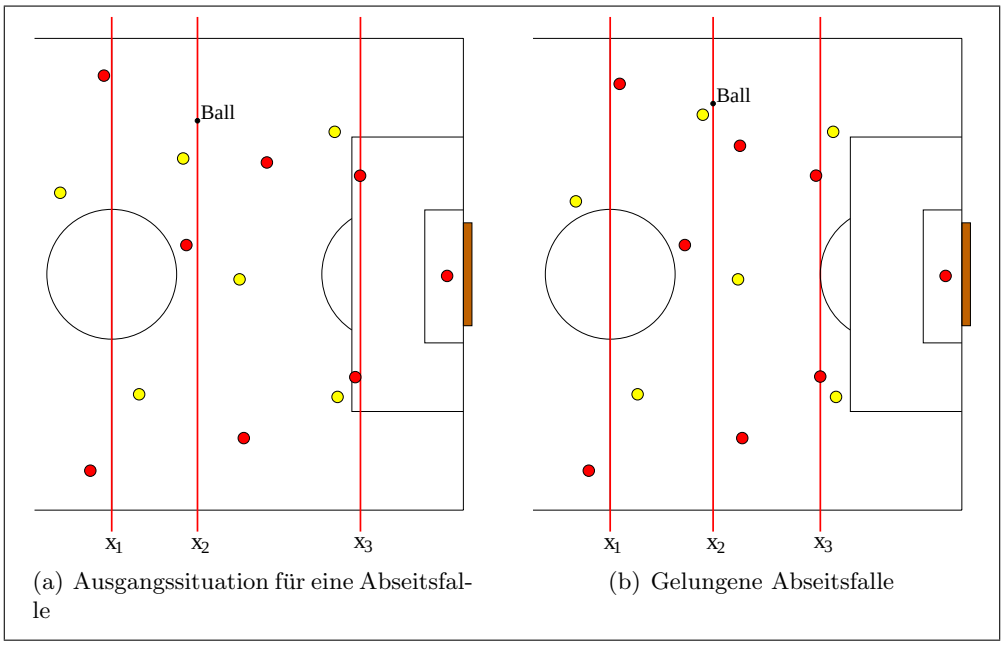
1. Sie befindet sich immer in der Spielhälfte des verteidigenden Teams und ist damit durch die Mittellinie begrenzt (dargestellt durch die Linie x_1 in Abbildung 4.10 auf der nächsten Seite).

2. Sie wird durch die Ballposition insofern eingegrenzt, daß alle Positionen zwischen der Ballposition und der Mittellinie sich nicht im Abseits befinden (dargestellt durch die Linie x_2 in Abbildung 4.10).

3. Sie ist weiterhin durch die Position des vorletzten Spielers der verteidigenden Mannschaft begrenzt, alle Positionen zwischen dem vorletzten Spieler und der Torlinie der verteidigenden Mannschaft sind Abseitspositionen (dargestellt durch die Linie x_3 in Abbildung 4.10).

Diejenige Linie der Linien x_1, x_2 und x_3, die sich am nächsten zur Torlinie der verteidigenden Mannschaft befindet, bildet die Abseitslinie, in Abbildung 4.10 ist sie also durch die Linie x_3 gegeben.

Im folgenden wird davon ausgegangen, daß der Torwart der letzte Spieler ist. Eine Abseitsfalle wird als koordinierte Aktion der verteidigenden Spieler



(a) Ausgangssituation für eine Abseitsfalle

(b) Gelungene Abseitsfalle

Abbildung 4.10: Beobachtungen von Start und Ende einer gelungenen Abseitsfalle zweier Verteidiger

betrachtet, die versuchen die Abseitslinie so zu verschieben, daß gegnerische Spieler in Abseitspositionen gebracht werden.

Damit ist die erste Bedingung für eine Abseitsfalle, daß die Spieler in der Verteidigung sich geschlossen in Richtung gegnerisches Tor bewegen. Die Linie x_3 aus Abbildung 4.10 ist dabei durch die Position des Verteidigers gegeben, der die geringste Entfernung zur eigenen Torlinie aufweist. Alle Spieler, die sich an diesem Verhalten beteiligen, müssen eine ähnliche Entfernung zur eigenen Torlinie aufweisen, damit von einem koordinierten Verhalten gesprochen werden kann. Diese **koordinierte Bewegung einer Gruppe** von verteidigenden Spielern P_1 im Zeitintervall von t_0 bis t_1 wird durch den Ausdruck

$$\text{groupMovement}\,(P_1, t_0, t_1, lastOwnDefenderPos),$$

dargestellt, wobei $lastOwnDefenderPos \in \mathbb{V}$ die Position des Verteidigers mit der geringsten Entfernung zur eigenen Torlinie zum Zeitpunkt t_1 bezeichnet.

Für eine erfolgreiche Abseitsfalle muß nach dem Vorrücken der Spieler-Gruppe P_1 einen nichtleere **Menge gegnerischer Spieler** sich **in Abseitspositionen** befinden, also hinter dem letzten Verteidiger. Diese Bedingung wird mit dem Ausdruck

$$\text{offsidePosition}\,(P_2, t_2, t_n)$$

dargestellt. P_2 bezeichnet dabei eine nichleere Menge von gegnerischen Spielern, die sich im Zeitintervall von Zeittakt t_2 bis t_n in einer Abseitsposition befinden.

Die **Abseitslinie** muß während des ganzen Zeitraums **durch die Verteidiger gegeben** sein, d.h. die Linie x_3 muß sich näher zur Torlinie der Mannschaft der Spielergruppe P_1 befinden als die Linien x_1 oder x_2. Wir kennzeichnen diese

Bedingung mit dem Ausdruck

$$\text{offsideLineGivenByDefender}\,(P_1, t_0, t_n),$$

wobei P_1 die Gruppe der verteidigenden Spieler, t_0 der Startzeitpunkt, und t_n der Endzeitpunkt des Verhaltens-Muster Abseitsfalle ist. Mit diesem Ausdruck wird also auch ausgesagt, daß eine Abseitsfalle nicht möglich ist, wenn sich der Ball bereits hinter dem letzten Verteidiger befindet oder die Verteidiger alle bis vor die Mittellinie vorrücken.

Für den Abschluß einer wirkungsvollen Abseitsfalle ist es wichtig, daß zumindest einer der **gegnerischen Spieler** die **im Abseits** stehen auch **nahe genug am Spielgeschehen** ist. Im Gegensatz zum richtigen Fußball kann im verwendeten Anwendungsszenario der Ball meist nur wenig mehr als 30 Meter gepasst werden. Befindet sich der Ball z.B. im gegnerischen Strafraum, ist eine Abseitsfalle nicht sinnvoll. Desweiteren muß ein **gegnerischer Spieler** die **Kontrolle über** den **Ball** haben, damit die Abseitsfalle erfolgreich abgeschlossen werden kann. Diese zwei Bedingungen sind im Ausdruck

$$\text{ballNearOffsideOpponents}\,(P_2, t_n)$$

zusammengefasst, wobei P_2 die sich im Zeittakt t_n im Abseits befindenden Menge von gegnerischen Spielern bezeichnet.

Die Spielermengen P_1 und P_2 müssen verständlicherweise aus Spielern **entgegengesetzter Teams** bestehen. Wir bezeichnen dies mit dem Ausdruck

$$\text{differentTeams}\,(P_1, P_2).$$

Abseitsfallen finden im Spielzustand *play on* statt, dürfen aber in einem anderem Spielzustand starten. Wir bezeichnen diese Bedingung wie gewohnt mit dem Ausdruck

$$\text{virtuallyPlayOn}\,(t_0, t_n).$$

Damit ergibt sich das Verhaltens-Muster Abseitsfalle (Verhaltens-Muster 9).

Verhaltens-Muster 9: Abseitsfalle der Spielergruppe P_1

$$\begin{aligned}&\text{offsideTrap}\,(P_1, P_2, t_0, t_n, lastOwnDefenderPos) :- \\ &\qquad \text{groupMovement}\,(P_1, t_0, t_1, lastOwnDefenderPos), \\ &\qquad \text{offsidePosition}\,(P_2, t_2, t_n),\ \text{follow}\,(t_1, t_2), \\ &\qquad \text{offsideLineGivenByDefender}\,(P_1, t_0, t_n),\ \text{virtuallyPlayOn}\,(t_0, t_n), \\ &\qquad \text{ballNearOffsideOpponents}\,(P_2, t_2, t_n),\ \text{differentTeams}\,(P_1, P_2).\end{aligned}$$

In Abbildung 4.10 auf der vorherigen Seite ist eine Ausgangssituation für eine Abseitsfalle (Teilbild (a)) und die erfolgreiche Beendingung einer Abseitsfalle (Teilbild (b)) zu sehen. Die Abseitsfalle wurde hier von zwei Spielern durchgeführt und zwei gegnerische Spieler wurden in eine Abseitspositionen gebracht von denen ein Spieler (der obere) ein attraktiver Anspielpartner gewesen wäre, stände er nicht im Abseits.

4.5 Objektorientierte Realisierung

In diesem Kapitel wurden bisher verschiedene Verhaltens-Muster eingeführt und jeweils durch eine Sequenz von Ereignis-Mustern beschrieben. Desweiteren wurde allgemein beschrieben, wie Verhalten dieser Verhaltens-Muster in Beobachtungssequenzen erkannt werden können. In diesem Abschnitt wird eine effiziente objektorientierte Realisierung des Prozesses der Verhaltens-Bestimmung vorgestellt. Ereignisse, Ereignis-Muster, Verhalten und Verhaltens-Muster werden objektorientiert spezifiziert und die Erkennung von Verhalten wird durch die Angabe von Algorithmen genau beschrieben.

Zuerst werden grundlegende Anforderungen an eine Realisierung gestellt und die getroffenen Entscheidungen zur Verwirklichung der Anforderungen erläutert (Abschnitt 4.5.1). Durch die getroffenen Entscheidungen ergeben sich u.a. verschiedene Verhaltenszustände, die eingeführt und beschrieben werden (Abschnitt 4.5.2). Danach wird die objektorientierte Realisierung von Ereignissen und Ereignis-Mustern mit ihren möglichen Zuständen beschrieben (Abschnitt 4.5.3). Im Anschluß untersuchen wir die verschiedenen Ereignis-Muster hinsichtlich ihrer Benutzung in Verhaltens-Mustern und erhalten eine Strukturierung der Ereignis-Muster (Abschnitt 4.5.4). Dann werden Verhalten und Verhaltens-Muster objektorientiert realisiert (Abschnitt 4.5.5), bevor schließlich ein Algorithmus zur Verhaltens-Erkennung angegeben wird (Abschnitt 4.5.6).

4.5.1 Grundlegende Anforderungen und Entscheidungen

Ausgehend von gestellten Anforderungen werden in diesem Abschnitt die wichtigsten getroffenen Entscheidungen für die Realisierung der Verhaltens-Bestimmung erläutert. Obwohl einige der Entscheidungen bereits durch das vorgestellte Konzept in Teilen vorweggenommen wurden, werden sie hier im Kontext der Anforderungen erklärt.

4.5.1.1 Anforderungen

Es werden drei grundlegende Anforderungen an die Realisierung gestellt.

- Die Verhaltens-Erkennung soll **effizient** sein, damit auch bei einer gleichzeitigen Untersuchung vieler Verhaltens-Muster die Verhalten noch in Echtzeit erkannt werden können. Effizient bedeutet, daß der Coach nur jeweils einen geringen Bruchteil eines Taktes mit der Verhaltens-Erkennung beschäftigt sein soll.

- Die Verhaltens-Muster müssen voneinander **unabhängig** sein, d.h. Änderungen an einem Verhaltens-Muster dürfen keine Änderungen anderer Verhaltens-Muster notwendig machen. Ebenso darf die Aufnahme eines zusätzlichen Verhaltens-Musters keine Modifikationen bereits existierender Verhaltens-Muster erfordern.

- Die erkannten Verhalten sollen **abgeschlossen** sein, d.h. es soll jeweils das längste Verhalten ermittelt werden, welches das Verhaltens-Muster noch erfüllt.

4.5.1.2 Entscheidungen

Zur Erfüllung der im letzten Abschnitt gestellten Anforderungen wurden mehrere Entscheidungen getroffen, die im folgenden motiviert und erörtert werden.

Jedes Verhaltens-Muster mit allen Ausprägungen wird durch eine Klasse definiert: Die Unabhängigkeit der verschiedenen Verhaltens-Muster untereinander wird durch eigene Klassen für jedes Verhaltens-Muster realisiert. Jede dieser Klassen implementiert dabei die abstrakte Klasse **Behavior** und beschreibt das jeweilige Verhaltens-Muster. Die Objekte dieser Klassen stellen sowohl vollständig beobachtete Verhalten, als auch nur in Anfängen beobachtete Verhalten dar. Zur Vermeidung von doppelten Berechnungen umfaßt dabei jede Verhaltens-Klasse alle Ausprägungen des entsprechenden Verhaltens-Musters. Das Verhaltens-Muster Paß wird z.B. durch die Klasse **PassBehavior** realisiert, welche u.a. die Eigenschaft hat, alle Paß-Ausprägungen zu bestimmen.

Jedes Verhaltens-Muster ist durch eine Sequenz von Ereignis-Mustern definiert. Die Ereignis-Muster selbst sind dabei auch durch verschiedene Klassen realisiert, die alle die abstrakte Klasse **Event** implementieren. Die Objekte dieser Klassen beschreiben sowohl vollständig beobachtete Ereignisse, als auch nur in Anfängen beobachtete Ereignisse. Ein Verhaltens-Objekt besteht damit aus Ereignis-Objekten.

Die Klassen für die Ereignis-Muster sind in Abschnitt 4.5.3 auf Seite 87, die Klassen für die Verhaltens-Muster in Abschnitt 4.5.5 auf Seite 94 beschrieben.

Sequentielle Arbeitsweise der Verhaltens-Erkennung: Um alle Verhalten eines Verhaltens-Muster für eine Beobachtungssequenz ermitteln zu können, müssen alle Beobachtungen mindestens einmal analysiert werden. Für eine effiziente Bestimmung sollte jede Beobachtung genau einmal betrachtet werden. Die Beobachtungen sollen dabei in ihrer natürlichen Reihenfolge untersucht werden.

Der Algorithmus der Verhaltens-Erkennung soll also sequentiell arbeiten, d.h. jeweils nur die aktuelle Beobachtung auswerten. Nach der Initialisierung eines Verhaltens-Objektes wird mit jeder neuen Beobachtung überprüft, ob das Verhalten stattfindet oder nicht. Damit durchläuft jedes Verhaltens-Objekt eine Art Lebenszyklus, der durch unterschiedliche Zustände in Abschnitt 4.5.2 auf der nächsten Seite strukturiert und damit auch genauer beschrieben wird.

Mit der sequentiellen Arbeitsweise steigt die Laufzeit der Verhaltens-Erkennung linear mit der Länge der untersuchten Beobachtungssequenz und linear mit der Anzahl der betrachteten Verhaltens-Muster. Der konkrete Algorithmus zur Verhaltens-Erkennung wird in Abschnitt 4.5.6 auf Seite 96 vorgestellt.

Strukturierung von Ereignis-Mustern: Bei der Erkennung eines Verhaltens muß im ungünstigsten Fall die Gültigkeit jedes im Verhaltens-Muster enthaltenen Ereignis-Musters für jede einzelne Beobachtung überprüft werden. Durch eine Strukturierung von Ereignis-Mustern innerhalb eines Verhaltens-Musters in Ereignisse die immer gelten müssen (z.B. Ereignis virtuallyPlayOn beim Paß) und in Ereignisse die nur zu gewissen Zeitpunkten gelten müssen

(z.B. Ereignis ballFastDeparting beim Paß), lassen sich diese Gültigkeitstests reduzieren. Damit muß die aktuelle Beobachtung nur noch mit den jeweils für die Verhaltens-Erkennung notwendigen Ereignis-Mustern verglichen werden. Für die Strukturierung von Ereignis-Mustern werden die Gemeinsamkeiten und Unterschiede der Ereignis-Muster in Abschnitt 4.5.4 auf Seite 90 analysiert.

Finale Erkennung eines Verhaltens: Bei einigen Verhalten werden wir mit einem interessanten Aspekt konfrontiert. Für ein längeres Dribbel-Verhalten z.B. erfüllen auch alle Teilsequenzen mit einer Mindestdauer von 4 Takten (Bedingung $\text{minTime}_4\,(t_0, t_n)$) dieses Verhaltens-Muster. Wir interessieren uns jedoch nur für das längste Dribbel-Verhalten, welches noch alle Bedingungen erfüllt.

Um das jeweils längste Verhalten zu ermitteln, welches dem Verhaltens-Muster noch entspricht, wird das untersuchte Verhalten solange mit neuen Beobachtungen überprüft, bis es in der aktuellen Beobachtung nicht mehr erfüllt ist bzw. bei der nächsten Beobachtung definitiv nicht mehr gelten wird. Dazu werden verschiedene Zustände von Verhalten (Abschnitt 4.5.2) benötigt.

4.5.2 Verschiedene Zustände von Verhalten

Durch die sequentielle Untersuchung einer Beobachtungsfolge auf stattfindendes Verhalten, durchläuft jedes Verhaltens-Objekt eine Art Lebenszyklus. Dieser Lebenszyklus ist in verschiedene Phasen strukturiert. Die jeweils aktuelle Phase wird durch den Zustand des Verhaltens beschrieben.

Bei der Erzeugung eines Verhaltens-Objektes wird sein Zustand initialisiert. Mit jeder analysierten Beobachtung kann der Zustand des Verhaltens wechseln, bis schließlich durch die Analyse einer Beobachtung festgestellt wird, daß das Verhalten beendet oder abgebrochen wurde.

Die möglichen Zustände von Verhaltens-Objekten werden in diesem Abschnitt vorgestellt und beschrieben. Desweiteren wird durch die Analyse der möglichen Zustandswechsel eines Verhaltens-Objektes der Lebenszyklus eines Verhaltens-Objektes veranschaulicht.

Bei der Überprüfung, ob ein Verhalten stattfindet oder nicht, wird der jeweils aktuelle Zustand des Verhaltens ermittelt. Insgesamt werden die folgenden sechs Zustände, in denen ein Verhalten auftreten kann, unterschieden.

***not happening* (dt. nicht stattfindend)** Ein Verhalten kann nicht stattfinden. Dies ist der Fall, wenn der Startzeitpunkt des Verhaltens noch nicht ermittelt werden konnte. Mit diesem Zustand wird jedes Verhaltens-Objekt initialisiert[20].

Ein Verhaltens-Objekt Paß befindet sich solange in diesem Zustand, bis in einer Beobachtung der Ball durch einen Interaktionspartner kontrolliert wird. Erst dann kann die Startzeit des Verhaltens ermittelt werden.

[20] Wann welche Verhaltens-Objekte angelegt werden wird später in Abschnitt 4.5.6.2 auf Seite 98 behandelt.

***pending* (dt. anhängig)** Ein Verhalten ist anhängig oder schwebend, wenn der Startzeitpunkt des Verhaltens schon ermittelt wurde, das Verhalten aber im eigentlichen Sinne noch nicht gestartet wurde. Ein Verhalten ist erst dann *im eigentlichen Sinne gestartet*, wenn eine ausreichende Teilmenge der Ereignisse – die sogenannten Vorbedingungen[21] – des Verhaltens bestimmt wurde.

Ein Verhaltens-Objekt Paß befindet sich z.B. nach der Untersuchung der ersten Beobachtungen, in der der Kicker allein den Ball kontrolliert, in diesem Zustand. Wenn bei der Untersuchung der zweiten Beobachtung festgestellt wird, daß der Ball von keinem Spieler kontrolliert wird (Ereignis-Muster ballFree erfüllt), der Ball sich aber nicht schnell vom Kicker entfernt (Ereignis-Muster ballFastDeparting nicht erfüllt), dann findet kein Paß-Verhalten statt und das Verhaltens-Objekt wird neu initialisiert.

***in progress* (dt. aktiv)** Ein Verhalten kann aktiv sein. Dies bedeutet, daß es im eigentlichen Sinne gestartet wurde aber noch kein möglicher Endzeitpunkt des Verhaltens festgestellt werden konnte, d.h. noch nicht alle Ereignisse des Verhaltens konnten bestimmt werden.

Ein erfolgreiches Paß-Verhalten ist also nach dessen eigentlichen Start solange aktiv, bis der Empfänger den Ball kontrolliert.

***end possible* (dt. Ende möglich)** Das Ende eines Verhaltens ist möglich, wenn alle Ereignisse des Verhaltens bestimmt wurden (möglicher Endzeitpunkt ermittelt), aber die prinzipielle Möglichkeit besteht, daß das Verhalten in der nächsten Beobachtung fortgesetzt wird.

Ein Dribbel Verhalten kann sich in diesem Zustand befinden sobald die minimale Dribbel-Dauer erreicht wurde.

***finished* (dt. beendet)** Ein Verhalten gilt als beendet, wenn mit der aktuellen Beobachtung das Verhalten nicht weiter fortgesetzt werden kann und der vorherige Zustand des Verhaltens-Objektes der Zustand *end possible* war.

Wenn z.B. bei der Untersuchung einer neuen Beobachtung für ein Dribbel-Objekt welches sich im Zustand *end possible* befindet, festgestellt wird, daß sich die Dribble-Richtung stark verändert hat (straightMovement-Bedingung nicht mehr erfüllt), so gilt das aktuelle Dribbel-Verhalten als beendet.

Ein Verhalten gilt auch als beendet, wenn alle Ereignisse des Verhaltens bestimmt wurden und das Verhalten mit der nächsten Beobachtung definitiv nicht fortgesetzt werden kann.

Dies ist z.B. beim Paß-Verhalten der Fall, wenn der Empfänger den Ball kontrolliert. Dann sind alle Ereignis-Muster bestimmt und das Verhalten kann durch die Bedingung XBallControl_1 nicht in der nächsten Beobachtung fortgesetzt werden.

[21] Die Vorbedingungen stellen eine Kategorie von Ereignissen dar und werden ausführlicher auf Seite 92 betrachtet.

***canceled* (dt. abgebrochen)** Ein Verhalten wird als abgebrochen bezeichnet, wenn das Verhalten bisher im Zustand *in progress* war und es mit der aktuellen Beobachtung nicht weiter fortgesetzt werden kann.

Ein Paß, der noch nicht den Empfänger erreicht hat aber im Zustand *in progress* ist, wird mit Beobachtungen abgebrochen, die einen Fehlerfall des Passes charakterisieren.

Bei der sequentiellen Untersuchung der Beobachtungssequenz aus Abbildung 4.4 auf Seite 52 durchläuft ein Paß-Objekt die Zustände *not happening* (Initialzustand), *pending* (Abb. 4.4(a)), *in progress* (Abb. 4.4(b-h)) und *finished* (Abb. 4.4(i)).

Der Zustand eines Verhaltens-Objektes wird durch ein Objekt der Klasse **State** repräsentiert (Abbildung 4.11).

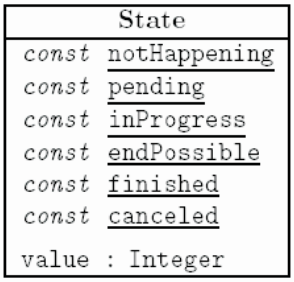


Abbildung 4.11: Klassendiagramm für die Klasse der Zustände

Alle möglichen Zustandsübergänge eines Verhaltens sind in Abbildung 4.12 mit Hilfe eines Petrinetzes dargestellt. Dabei wird durch jede neue Beobachtung eine Transition ausgelöst. Die Abhängigkeit des Schaltens einer Transition durch neue Beobachtungen ist im Petrinetz aus Gründen der Übersichtlichkeit nicht dargestellt.

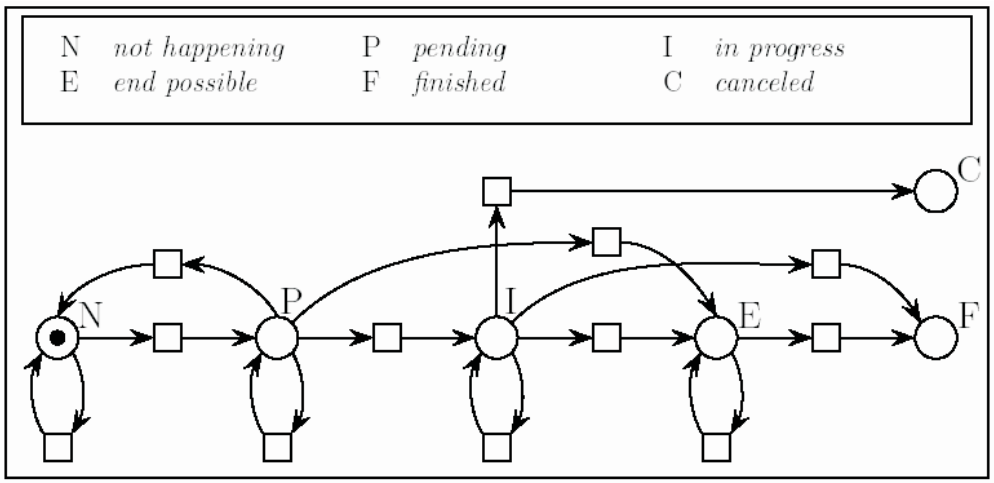


Abbildung 4.12: Petrinetz zur Beschreibung der Übergänge der Zustände eines Verhaltens

Am Anfang befindet sich das Verhalten im Zustand *not happening*. Durch die erste untersuchte Beobachtung kann der Zustand zu *pending* wechseln oder gleich bleiben.

Wenn das Verhaltens-Objekt im Zustand *pending* ist kann es mit einer neuen Beobachtung in den Zustand *in progress* wechseln aber auch ein Wechsel direkt in den Zustand *end possible* ist nicht ausgeschlossen. Erfolgreiches Dribbel-Verhalten mit ständiger Ballkontrolle wechselt z.B. vom Zustand *pending* in den Zustand *end possible* wenn die minimale Dribbel-Dauer erreicht ist. Das Verhalten kann auch im Zustand *pending* bleiben oder zurück in den Zustand *not happening* wechseln. Ein Paß-Verhalten wechselt z.B. vom Zustand *pending* in den Zustand *not happening*, wenn sich der Ball nach der Kontrolle durch einen Spieler von diesem nur langsam entfernt und nicht schnell, wie durch die Bedingung ballFastDeparting gefordert.

Aus dem Zustand *in progress* kann das Verhalten zum Zustand *end possible*, *canceled*, oder *finished* wechseln oder im aktuellen Zustand verharren. Ein Torschuß-Verhalten befindet sich nach Verlassen der Kontrolle des Kickers im Zustand *in progress* und wechselt z.B. in den Zustand *canceled* wenn der Ball vom Torwart abgefangen wird. Wenn der Ball eines Torschuß-Verhaltens dagegen das Tor trifft, wechselt der Zustand des Verhaltens von *in progress* zu *finished*.

Wenn das Verhaltens-Objekt bereits im Zustand *end possible* ist, kann es in diesem Zustand bleiben oder in den Zustand *finished* übergehen.

Zu Beginn wird für jede Verhaltens-Klasse, also für jedes Verhaltens-Muster ein Objekt angelegt, welches mit dem Zustand *not happening* initialisert wird. Dieses Objekt durchläuft einen Lebenszyklus bis es schließlich in den Zustand *finished* bzw. *canceled* gelangt. Damit ist ein Verhalten (erfolgreich bzw. erfolglos) erkannt und ein neues Verhalten dieses Verhaltens-Muster kann beginnen. Dafür wird ein weiteres Verhaltens-Objekt dieser Klasse angelegt.

Die Zustandsänderung eines Verhaltens-Objektes durch eine neue Beobachtung ist von den Zuständen seiner Ereignis-Objekte abhängig. Bevor genau angeben werden kann, wie sich der Zustand eines Verhaltens aus den Zuständen seiner Ereignisse zusammensetzt (Abschnitt 4.5.6), wird die Klasse der Ereignis-Muster mit ihren möglichen Zuständen eingeführt (Abschnitt 4.5.3) und die Bedeutungen der verschiedenen Ereignis-Muster hinsichtlich ihrer Benutzung in den Verhaltens-Mustern untersucht (Abschnitt 4.5.4).

4.5.3 Die Klassen für Ereignis-Muster

Die Ereignis-Klassen werden in diesem Abschnitt eingeführt. Dazu wird ihr Grundaufbau durch die Definition ihrer Superklasse **Event** vorgestellt (Abschnitt 4.5.3.1). Ereignis-Objekte durchlaufen genauso wie Verhaltens-Objekte eine Art Lebenszyklus, der durch die Angabe unterschiedlicher Zustände strukturiert wird (Abschnitt 4.5.3.2). Danach werden einige konkrete Ereignis-Klassen näher beschrieben. Die Bestimmung des Zustandes dieser Ereignis-Objekte wird basierend auf der aktuellen Beobachtung und dem letzten Zustand angegeben (Abschnitt 4.5.3.3).

4.5.3.1 Grundaufbau von Ereignis-Mustern

Die Hauptaufgabe eines Ereignis-Objektes ist die Analyse der aktuellen Beobachtung mit dem Ziel, den Zustand des Ereignisses zu bestimmen. Diese Aufgabe wird von der Methode *happens* erledigt, die den aktuellen Zustand als Rückgabewert liefert. Diese Methode aktualisiert außerdem die Start- und die Endzeit des Ereignisses, die mit eigenen Methoden (*getStartTime* bzw. *getEndTime*) erfragt werden können.

Die Superklasse aller Ereignis-Klassen ist durch die abstrakte Klasse **Event** gegeben. Die wesentlichen Eigenschaften dieser Klasse sind in Abbildung 4.13 als Klassendiagramm dargestellt. Die abstrakte Methode *happens* muß für jede abgeleitete Ereignis-Klasse definiert werden.

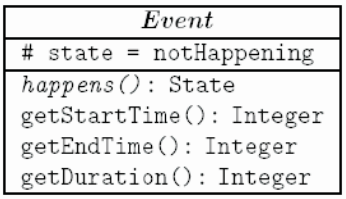


Abbildung 4.13: Klassendiagramm für die Klasse der Ereignisse

4.5.3.2 Zustände von Ereignissen

Die Ereignis-Objekte können auch in den sechs Zuständen *not happening*, *pending*, *in progress*, *end possible*, *finished* und *canceled* auftreten, wobei jedes konkrete Ereignis-Objekt nur in einigen dieser Zustände vorkommen kann.

Prinzipiell können Ereignis-Objekte in 3 Gruppen von Ereignis-Zuständen auftreten:

- ein Ereignis kann gelten (Zustände *end possible* und *finished*),
- ein Ereignis kann nicht gelten (Zustände *not happening* und *canceled*) und
- es kann sein, daß die Gültigkeit des Ereignisses noch nicht bestimmbar ist (Zustände *pending* und *in progress*).

In jeder dieser Gruppen unterscheiden sich die Ereignis-Zustände nur hinsichtlich ihrer Bedeutung die sie für ein Verhaltens-Objekt haben. Dies wird in den folgenden zwei Absätzen mit Beispielen erläutert.

So haben die Bedingungen sameTeam und notSame unterschiedliche Bedeutungen für das Verhaltens-Objekt Paß wenn sie nicht gelten. Wenn das Ereignis sameTeam nicht gilt, bedeutet dies, daß der Gegner die Kontrolle über den Ball erlangt hat und damit der Paß fehlgeschlagen ist (Ereignis-Zustand *canceled*). Wenn dagegen das Ereignis notSame nicht gilt, bedeutet dies, daß der Kicker den Ball wieder kontrolliert, also gar kein Paß-Verhalten sondern ein Klären-Verhalten stattgefunden hat (Ereignis-Zustand *not happening*). Damit haben die Zustände *canceled* und *not happening* eine unterschiedliche Bedeutung.

Ähnliches gilt für die beiden Zustände *pending* und *in progress* die beide ausdrücken, daß die Gültigkeit des Ereignisses erst zu einem späteren Zeitpunkt ermittelt werden kann. Sie unterscheiden sich aber hinsichtlich ihrer Bedeutung für ein Verhaltens-Objekt. Der Zustand *in progress* bedeutet, daß das Verhalten

aus der Sicht des Ereignisses im eigentlichen Sinne gestartet wurde, wohingegen der Zustand *pending* das Gegenteil aussagt. Das Ereignis minTime_4 (eine Bedingung des Dribbel-Verhalten) befindet sich in den ersten drei Zeittakten im Zustand *pending*. Damit gilt ein Dribbel-Verhalten in den ersten drei Zeittakten als noch nicht im eigentlichen Sinne gestartet. Das Ereignis $\text{sameTeam}\,(p_1, p_2)$ (eine Bedingung des Paß-Verhalten) befindet sich bis die Spieler p_1 und p_2 bestimmt sind im Zustand *in progress*. Damit gilt ein Paß-Verhalten für diesen Zeitraum aus Sicht dieses Ereignisses als im eigentlichen Sinne gestartet. Dies ändert aber nichts an der Tatsache, daß die Zustände der anderen Ereignisse eines Verhaltens ebenso zur Bestimmung des Verhaltens-Zustandes beitragen. Wie die Zustände der Ereignisse kombiniert werden um den Verhaltens-Zustand zu ermitteln, wird erst später in Abschnitt 4.5.6.1 auf Seite 96 beschrieben.

Die Zustände haben bei den Ereignis-Objekten teilweise andere Bedeutungen als bei den Verhaltens-Objekten. Deshalb werden sie nochmals kurz erläutert.

not happening Ein Ereignis findet nicht statt, wenn die Bedingung in der aktuellen Beobachtung nicht gilt. Dieser Zustand ist der Initialzustand aller Ereignisse.

in progress Mit diesem Zustand wird ausgesagt, daß die Gültigkeit der Bedingung des Ereignisses erst zu einem späteren Zeitpunkt festgestellt werden kann.

pending Dieser Ereignis-Zustand bedeutet dasselbe wie der Zustand *in progress* mit dem Unterschied, daß in diesem Zustand die Verhalten, die mit diesem Ereignis definiert sind, als nicht im eigentlichen Sinne gestartet angesehen werden.

end possible Durch diesen Zustand wird ausgedrückt, daß die Bedingung des Ereignisses in der aktuellen Beobachtung gilt, sie aber prinzipiell auch noch in der nächsten Beobachtung gelten kann. Das Ereignis ballFree befindet sich z.B. in diesem Zustand, wenn der Ball von keinem Spieler kontrolliert wird.

finished Ein Ereignis ist beendet, wenn es entweder in der aktuellen Beobachtung gilt und nicht in der nächsten Beobachtung gelten kann. Desweiteren ist es beendet, wenn das Ereignis mit der aktuellen Beobachtung nicht mehr fortgesetzt werden kann, aber in der vorangehenden Beobachtung im Zustand *end possible* war. In diesem Zustand befindet sich z.B. das Ereignis XBallControl_1, wenn genau ein Spieler den Ball kontrolliert.

canceled Dieser Zustand sagt aus, daß die Bedingung des Ereignisses in der aktuellen Beobachtung nicht gilt. Im Gegensatz zum Zustand *not happening* gilt hier das Verhalten, welches mit einem solchen Ereignis definiert wurde bereits als im eigentlichen Sinne gestartet.

4.5.3.3 Einige konkrete Ereignis-Klassen

Für jede Bedingung (Ereignis-Muster) der eingeführten Verhaltens-Muster wird eine Ereignis-Klasse definiert, die von der Klasse **Event** abgeleitet ist. In Abbildung 4.14 werden beispielhaft die vier Ereignis-Klassen **BallFastDeparting**, **XBallControl2**, **SameTeam** und **MinTime4** näher vorgestellt. Diese Klassen geben die Ereignis-Muster ballFastDeparting, XBallControl_2, sameTeam bzw. minTime_4 objektorientiert wieder.

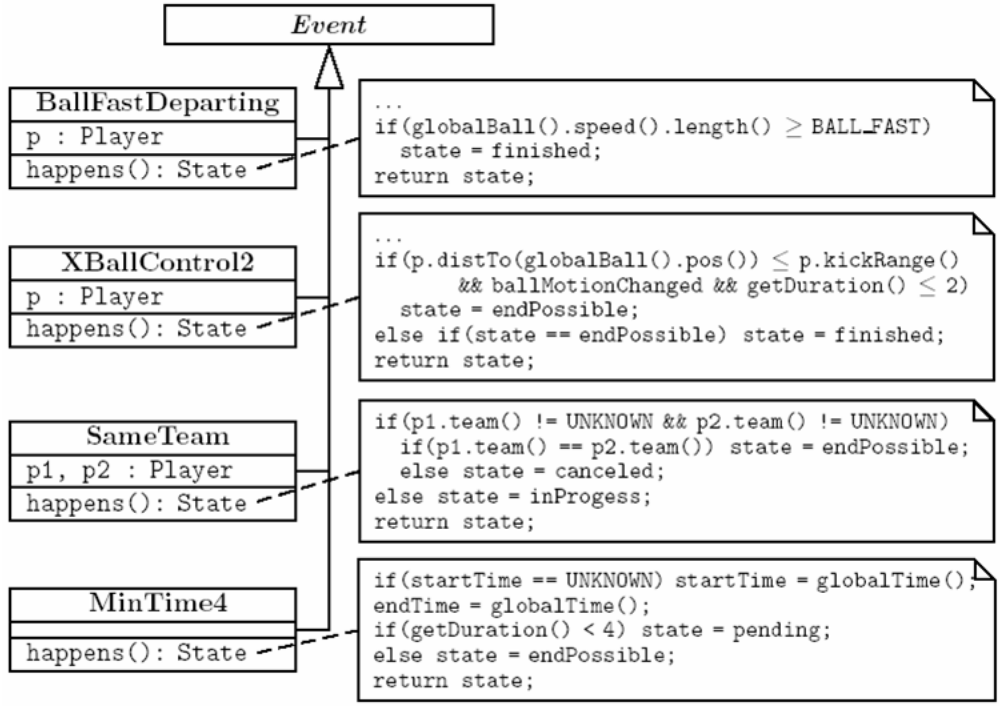


Abbildung 4.14: Klassendiagramm für einige Ereignis-Klassen

In diesem Klassendiagramm sind die gemeinsamen Attribute mit dem übergeordneten Verhaltens-Objekt dargestellt (p, p1 und p2). Die entsprechenden Referenzen auf diese Attribute werden bei der Erzeugung des jeweiligen Ereignis-Objektes übergeben.

Die Hauptfunktionalität der vier Ereignis-Klassen ist durch die Methode *happens* bestimmt. Diese Methode ist in der Abbildung jeweils in ihren wesentlichen Zügen angegeben, für die Klassen **SameTeam** und **MinTime4** sogar vollständig.

4.5.4 Strukturierung der Ereignisse bzgl. ihrer Nutzung

In diesem Abschnitt geht es um die Ereignis-Objekte im Kontext ihrer Benutzung in den Verhaltens-Objekten. Es werden die Gemeinsamkeiten und Unterschiede der Ereignis-Objekte eines beliebigen Verhaltens-Musters untersucht und die Ereignis-Objekte in Kategorien gruppiert.

Die Ereignis-Muster werden in Hinsicht ihrer unterschiedlichen Nutzung im Verhaltens-Muster klassifiziert. Ein Ereignis-Muster spezifiziert dabei eine konkrete Art von Bedingung. Es werden die folgenden Bedingungen unterschieden:

1. **Zeitliche Bedingungen**
   Mit den zeitlichen Bedingungen (follow (t_j, t_k)) wird eine lückenlose Abfolge einiger anderer Ereignis-Muster definiert. Die Abfolge dieser anderen Ereignis-Muster wird durch eine Ordnung innerhalb der Kategorie der Sequenz-Bedingungen beschrieben, so daß die Ereignis-Muster, die zeitliche Bedingungen ausdrücken, nicht als Ereignis-Objekte implementiert werden müssen.

2. **Sequenz-Bedingungen**
   Die Ereignis-Objekte dieser Kategorie definieren eine Ereignis-Abfolge vom Start bis zum Ende des Verhaltens und sind total geordnet, d.h. jedes außer das letzte Ereignis-Objekt dieser Kategorie wird durch ein anderes Ereignis-Objekt dieser Kategorie abgelöst. Die Ordnung ist dabei durch die zeitlichen Bedingungen und durch gemeinsame Zeitpunkt-Variablen definiert. Wenn das aktuell untersuchte Ereignis-Objekt dieser Kategorie den Zustand *finished* erreicht hat, wird das darauf folgende Ereignis-Objekt mit der aktuellen Beobachtung untersucht. Die geordneten Ereignis-Objekte dieser Kategorie werden in einer Liste (Datenstruktur) mit dem Namen *sequenceEvents* zusammengefasst.

   Für das Paß-Verhalten besteht diese Ereignis-Abfolge aus den Bedingungen 1. XBallControl_2, 2. ballFastDeparting, 3. teammateInKickRegion, 4. ballFree und 5. XBallControl_1.

   Die Sequenz-Bedingungen können weiter unterteilt werden und zwar in Zeitpunkt-Bedingungen, Intervall-Bedingungen und Vorbedingungen.

   (a) **Zeitpunkt-Bedingungen**
   gelten entweder in einem Zeitpunkt oder nicht, d.h. der Zustand einer Zeitpunkt-Bedingung ist immer mit genau einer Beobachtung feststellbar. Die möglichen Zustände für solche Ereignis-Objekte sind *not happening* und *finished*. Alle anderen Zustände sind nur für Zeitintervalle relevant.

   Ein Beispiel für eine solche Bedingung ist ballFastDeparting.

   (b) **Intervall-Bedingungen**
   weisen einen Start- und einen Endzeitpunkt auf, wobei die beiden Zeitpunkte auch übereinstimmen können. Bei Intervall-Bedingung wird der längstmögliche Zeitraum bestimmt, für den die Bedingung gilt. Dafür müssen solange Beobachtungen untersucht werden, bis die Bedingung nicht mehr gilt. Die möglichen Zustände für Ereignis-Objekte dieser Kategorie sind *not happening*, *in progress*, *end possible* und *finished*. Die anderen beiden Zustände *pending* und *canceled* werden infolge der Ähnlichkeit zu den Zuständen *in progress* bzw. *not happening* nicht benötigt. Die Bedeutungsverschiedenheit dieser Zustände (vgl. Abschnitt 4.5.3.2 auf Seite 88) wird bei den

Intervall-Bedingungen mit Hilfe von sogenannten Vorbedingungen realisiert.

Wenn ein Ereignis-Objekt dieser Kategorie im Zustand *in progress* oder *end possible* ist, wird es solange mit neuen Beobachtungen überprüft, bis es nicht weiter fortgesetzt werden kann (ausgedrückt durch die Zustände *not happening* bzw. *finished*).

Die Bedingung XBallControl_2 stellt ein Beispiel für eine Intervall-Bedingungen dar.

(c) **Vorbedingungen**[22]
sind die Bedingungen, die erfüllt sein müssen, damit im eigentlichen Sinne vom Start des Verhaltens gesprochen werden kann. Durch die Ordnung der Sequenz-Bedingungen sind alle Ereignis-Objekte der Sequenz-Bedingungen bis zu einem bestimmten Ereignis-Objekt Vorbedingungen. Die folgenden Ereignis-Objekte sind dann keine Vorbedingungen mehr. Sobald also die erste Nicht-Vorbedingung untersucht wird, gilt das Verhalten als *im eigentlichen Sinne gestartet*. Demnach reicht es aus, wenn die Nummer der ersten Nicht-Vorbedingung für jedes Verhalten angeben wird. Diese Nummer wird mit *inProgressEventNum* bezeichnet.

Die erste Nicht-Vorbedingung des Paß-Verhaltens ist die Bedingung ballFree (4. Sequenz-Bedingung), so daß das Attribut *inProgressEventNum* des Verhaltens-Objektes Paß den Wert 4 hat.

3. **Immer-Bedingungen**
Die Immer-Bedingungen stellen die Ereignis-Muster eines Verhaltens-Musters dar, die vom Startzeitpunkt (t_0) bis zum Endzeitpunkt (t_n) des Verhaltens gelten müssen. Die Ereignis-Objekte dieser Kategorie werden unabhängig voneinander ausgewertet und können in allen sechs Zuständen vorkommen. Wenn eines dieser Ereignis-Objekte einen der Zustände *not happening*, *finished* oder *canceled* annimmt, bedeutet dies, daß auch das Verhalten nicht fortgesetzt werden kann. Alle Ereignis-Objekte dieser Kategorie sind in einer Menge mit dem Namen *alwaysEvents* zusammengefasst.

Die Immer-Bedingungen des Verhaltens-Muster Paß sind die Bedingungen sameTeam, notSame und virtuallyPlayOn.

Auch diese Kategorie läßt sich weiter untergliedern und zwar in feststehende und veränderbare Immer-Bedingungen.

(a) **Feststehende Immer-Bedingungen**
ändern ihre Ausprägung über den ganzen Zeitraum des Verhaltens nicht, d.h. wenn erst einmal bestimmt wurde, ob die Bedingung gilt oder nicht, dann ändert sich der Zustand von feststehenden Immer-Bedingungen nicht mehr. Solange die Gültigkeit der Bedingung nicht feststellbar ist, hat sie den Zustand *in progress*. Wenn die Gültigkeit

[22] Die Vorbedingungen eines Verhaltens werden ebenso wie das Verhaltens-Muster selbst durch einen Designer bestimmt.

ermittelbar ist, findet ein Wechsel in einen der Zustände *end possible*, *not happening* oder *canceled* statt, der sich dann nicht mehr ändert. Ein Wechsel in den Zustand *finished* ist nicht möglich, da die Bedingungen dieser Kategorie ja immer gelten und demnach nicht beendet sein können.

Die Bedingung sameTeam ist ein Beispiel-Ereignis dieser Kategorie. Sobald beide Spieler feststehen, kann die Gültigkeit bestimmt werden.

(b) **Veränderbare Immer-Bedingungen**
sind nicht auf eine Ausprägung für den ganzen Zeitraum des Verhaltens festgelegt. Sie können prinzipiell in allen sechs Zuständen auftreten und mehrmals ihren Zustand wechseln. Bisher wurden nur die vier Zustände *not happening*, *pending*, *end possible* und *finished* für veränderbare Immer-Bedingungen benötigt.

Ein Beispiel für eine solche Bedingung ist virtuallyPlayOn.

4. **Parallel-Bedingungen**
Mit den Ereignis-Objekten dieser Kategorie können weitere Bedingungen an die Sequenz-Bedingungen gekoppelt werden. Der Start- und Endzeitpunkt einer Parallel-Bedingung kann dabei nur an die verschiedenen Start- und Endzeitpunkte der Sequenz-Bedingungen angeknüpft werden. Bei gewünschten Überlappungen mit einzelnen Sequenz-Bedingungen muß die entsprechende Sequenz-Bedingung vorher entsprechend zerlegt werden. Damit stellen die Immer-Bedingungen einen Spezialfall von Parallel-Bedingungen dar, wobei der Startzeitpunkt jeder Immer-Bedingung an den Startzeitpunkt der ersten Sequenz-Bedingung und der Endzeitpunkt jeder Immer-Bedingung an den Endzeitpunkt der letzten Sequenz-Bedingung geknüpft ist. Die Ereignis-Objekte dieser Kategorie werden in einer Menge mit dem Namen *parallelEvents* zusammengefasst.

   Das Verhaltens-Muster Paß hat mit der Bedingung movement genau eine Parallel-Bedingung.

5. **Abbruch-Bedingungen**
Wenn das Verhaltens-Objekt im Zustand *canceled* ist (es also nicht erfolgreich beendet wurde), werden mit den Abbruch-Bedingungen die Fehler-Ausprägungen des Verhaltens bestimmt. Eine Abbruch-Bedingung gilt entweder zu einem Zeitpunkt oder nicht. Die möglichen Zustände für Ereignis-Objekte dieser Kategorie sind demnach *not happening* (Abbruch-Bedingung gilt nicht) und *finished* (Abbruch-Bedingung gilt). Die Ereignis-Objekte dieser Kategorie werden durch eine Menge mit dem Namen *cancelEvents* repräsentiert.

   Eine Beispiel-Bedingung dieser Kategorie ist offside.

Diese Gruppierung wurde auf Basis der in dieser Arbeit vorgestellten Verhaltens-Muster vorgenommen. Die Ereignisse der betrachteten Verhalten können vollständig auf die obigen Gruppen verteilt werden.

In Abbildung 4.15 auf der nächsten Seite sind die Sequenz-Bedingungen, die Immer-Bedingungen und die Parallel-Bedingungen des Verhaltens-Muster

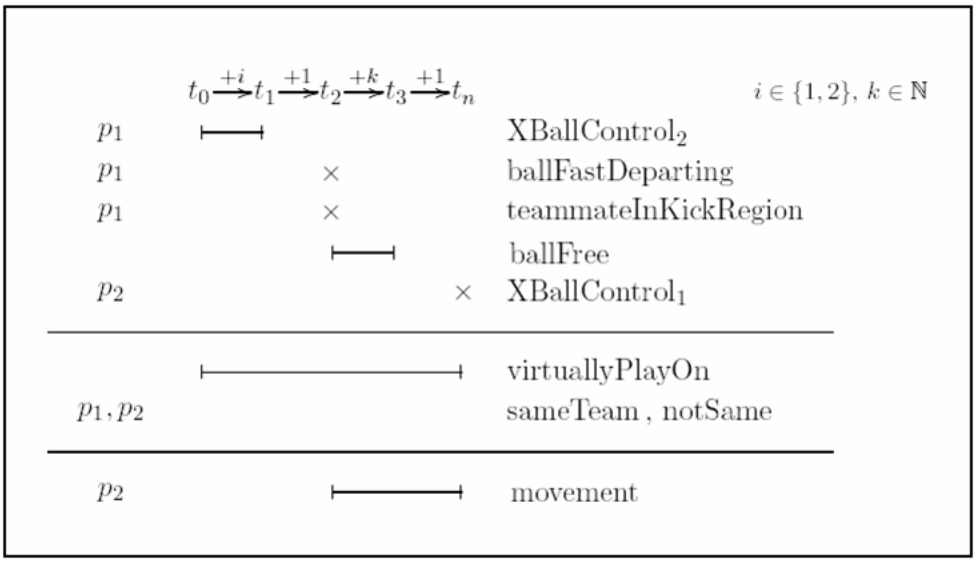


Abbildung 4.15: Graphische Darstellung des Verhaltensmuster Paß

Paß dargestellt. Im oberen Teil sind die Sequenz-Bedingungen zu sehen, die Intervall-Bedingungen sind dabei durch $\vdash\dashv$ und die Zeitpunkt-Bedingungen durch $\times$ dargestellt. Im mittleren Teil werden die Immer-Bedingungen gezeigt, veränderbare Immer-Bedingungen sind durch $\vdash\dashv$ dargestellt und feststehende Immer-Bedingungen sind nicht weiter gekennzeichnet. Im unteren Teil ist die Parallel-Bedingung des Verhaltens-Muster Paß zu sehen. Mit t_0, t_1, t_2, t_3 und t_n sind die aus der Definition des Verhaltens-Muster Paß verwendeten Zeitpunkte dargestellt. Die zeitliche Dauer zwischen diesen Zeitpunkten ist über den Pfeilen gekennzeichnet (z.B. $t_2 + k = t_3$).

4.5.5 Die Klassen für Verhaltens-Muster

In diesem Abschnitt wird der Aufbau der Verhaltens-Objekte vorgestellt. Die Bestandteile der abstrakten Superklasse **Behavior** werden beschrieben und es wird angegeben, welche Variablen und Methoden in den abgeleiteten Klassen definiert werden. Die Hauptbestandteile der Klasse **Behavior** und die davon abgeleiteten Klassen sind in Abbildung 4.16 auf der nächsten Seite als Klassendiagramm zusammengefaßt.

Die Verhaltens-Erkennung wird mit der Methode *happens* realisiert. Der Algorithmus dieser Methode ist von den konkreten Verhaltens-Klassen unabhängig. Deshalb ist die Methode *happens* bereits in der Klasse **Behavior** definiert. Der Algorithmus ermittelt den Zustand (*state*) des jeweiligen Verhaltens-Objektes, der sich aus den Zuständen der beteiligten Ereignis-Objekte zusammensetzt. Der Algorithmus wird in Abschnitt 4.5.6 auf Seite 96 vorgestellt.

Bei der Ermittlung des Zustandes werden desweiteren die Attribute (*values*) des Verhaltens-Objektes durch die Ereignis-Objekte bestimmt bzw. aktualisiert. Da die Attribute von Verhaltens-Muster zu Verhaltens-Muster verschieden sind, werden sie in den abgeleiteten Klassen definiert.

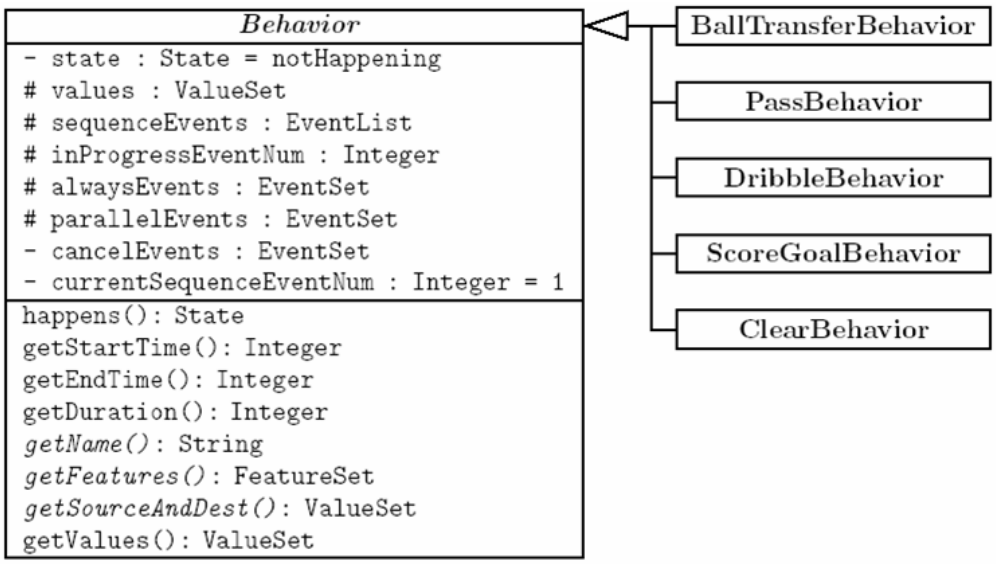


Abbildung 4.16: Klassendiagramm für die Verhaltens-Klassen

Für die Definition des von konkreten Verhaltens-Klassen unabhängigen Algorithmus zur Verhaltens-Erkennung wurden die Ereignis-Objekte bzgl. ihrer Nutzung in Kategorien unterteilt. Zur Definition eines Verhaltens-Objektes werden vier dieser Kategorien benötigt und zwar die Sequenz-Bedingungen, die Immer-Bedingungen, die Parallel-Bedingungen und die Abbruch-Bedingungen. Damit enthält jede Verhaltens-Klasse eine Liste mit dem Namen *sequenceEvents* und drei Mengen mit den Namen *alwaysEvents*, *parallelEvents* und *cancelEvents*. Desweiteren wurde die Variable *inProgressEventNum* spezifiziert um den Wechsel in den Zustand *in progress* feststellen zu können. Das aktuell untersuchte Ereignis-Objekt der *sequenceEvents* wird mit der Variablen *currentSequenceEventNum* gespeichert.

Die Kategorie der *cancelEvents* ist dabei für alle betrachteten Verhaltens-Klassen gleich und wird daher schon in der Klasse **Behavior** definiert. Die anderen 3 Kategorien und die Variable *inProgressEventNum* sind in den abgeleiteten Verhaltens-Klassen definiert.

Die Start- und die Endzeit eines Verhaltens-Objektes kann mit den Methoden *getStartTime* und *getEndTime* erfragt werden. Die Belegung aller Attribute eines Verhaltens-Objektes, und damit die konkrete Beschreibung des Verhaltens, wird durch die Methode *getValues* zurückgegeben. Die anderen Stufen der Beschreibung von Verhalten (vgl. Abschnitt 4.3.2 auf Seite 56) werden mit den Methoden *getName* (allgemeine Beschreibung), *getFeatures* (Ausprägungs-Beschreibung) und *getSourceAndDest* (Quelle-Ziel-Beschreibung) geliefert. Die Methoden *getName*, *getFeatures* und *getSourceAndDest* sind abstrakt und müssen für jede Verhaltens-Klasse definiert werden.

Von der abstrakten Klasse **Behavior** wurden die 5 Verhaltens-Klassen **BallTransferBehavior**, **PassBehavior**, **DribbleBehavior**, **ClearBehavior** und **ScoreGoalBehavior** abgeleitet und implementiert.

4.5.6 Erkennung von Verhaltens-Objekten

Nachdem die Verhaltens-Klassen eingeführt sind, wird in diesem Abschnitt beschrieben, wie Verhaltens-Objekte erkannt werden. Zuerst wird die Methode *happens* beschrieben (Abschnitt 4.5.6.1) und damit die Arbeitsweise des Algorithmus zur Verhaltens-Erkennung. Die Verhaltens-Klassen und ihre Funktionalität werden genutzt, um alle Verhalten einer Beobachtungssequenz zu erkennen (Abschnitt 4.5.6.2).

4.5.6.1 Algorithmus zur Verhaltens-Erkennung

Der – von konkreten Verhaltens-Klassen unabhängige – Algorithmus zur Verhaltens-Erkennung ist durch die Methode *happens* definiert, die den jeweils aktuellen Zustand des Verhaltens-Objektes ermittelt.

Dieser Zustand *state* setzt sich in jeder Beobachtung aus dem Zustand der Sequenz-Bedingungen, dem Zustand der Immer-Bedingungen und dem Zustand der Parallel-Bedingungen zusammen, d.h. bei der Auswertung des Zustandes mit der Methode *happens* wird jeweils ein Teilzustand für die *sequenceEvents*, für die *alwaysEvents* und für die *parallelEvents* bestimmt.

Diese drei Teilzustände werden dann zur Bestimmung des Gesamtzustandes des Verhaltens-Objektes kombiniert. Das jeweilige Ergebnis der Kombination zweier Teilzustände ist durch die Matrix in Tabelle 4.4 gegeben. Diese Matrix ist symmetrisch, damit ist die Kombinations-Operation kommutativ. Desweiteren ist die Kombinations-Operation assoziativ[23]. Folglich spielt die Reihenfolge von Kombinationen keine Rolle.

	N	P	I	E	F	C
not happening (N)	N	N	N	N	N	N
pending (P)	N	P	P	P	N	N
in progress (I)	N	P	I	I	C	C
end possible (E)	N	P	I	E	F	C
finished (F)	N	N	C	F	F	C
canceled (C)	N	N	C	C	C	C

Tabelle 4.4: Kombination zweier Zustände zu einem neuen Zustand

Wenn der Gesamtzustand den Wert *canceled* hat werden die Fehler-Ausprägungen des Verhaltens-Objektes mit Hilfe der Abbruch-Bedingungen bestimmt. Dabei können prinzipiell auch mehrere oder gar keine Abbruch-Bedingungen gelten. Damit ergibt sich der in Algorithmus 4.1 auf der nächsten Seite dargestellte Ablauf für die Verhaltens-Erkennung.

Die Bestimmung des Zustandes der Sequenz-Bedingungen (im folgenden mit Sequenz-Zustand bezeichnet), also die Abarbeitung der Methode *happensSequenceEvents*, ist im Algorithmus 4.2 auf der nächsten Seite dargestellt und wird im folgenden näher beschrieben.

[23] Die Assoziativität der Kombinations-Operation wurde durch einen vollständigen Test aller möglichen Kombinationen von Zuständen nachgewiesen.

Algorithmus 4.1: Grundaufbau der Verhaltens-Erkennung

```
//Auswertung der Sequenz-, Immer- und Parallel-Bedingungen
State sequenceState = happensSequenceEvents();
State alwaysState = happensAlwaysEvents();
State parallelState = happensParallelEvents( sequenceState );

//Ermittlung des Gesamtzustandes
state = combineStates( sequenceState, alwaysState );
state = combineStates( state, parallelState );

//Auswertung der Abbruch-Bedingungen
if( state == canceled ) happensCancelEvents();
```

Algorithmus 4.2: Auswertung der Sequenz-Bedingungen

```
State sequenceState = state; //bisheriger Verhaltenszustand
boolean checkEvent = YES;

while( checkEvent == YES ) {
  State eventState = sequenceEvents.elementAt( //Ereignis-Zustand
                       currentSequenceEventNum ).happens();
  checkEvent = NO;

  //Aktualisierung des Sequenz-Zustandes
  if( sequenceState == notHappening ) //Verhalten noch nicht gestartet
    if( eventState != notHappening ) sequenceState = pending;
  if( sequenceState == pending &&
      currentEventSequenceNum == inProgressEventNum )
    sequenceState = inProgress;
  if( sequenceState == inProgress && eventState == notHappening )
    sequenceState = canceled;

  if( currentSequenceEventNum == sequenceEvents.count() ) {
    //alle Sequenz-Bedingungen abgearbeitet
    if( eventState == endPossible ) sequenceState = endPossible;
    if( eventState == finished ) sequenceState = finished;
  }
  else {
    if( eventState == finished ) { //Ereigniswechsel
      checkEvent = YES; //Beobachtung mit folgenden Event untersuchen
      currentSequenceEventNum++;
    }
  }
}
```

Als Ausgangbasis wird der Sequenz-Zustand (*sequenceState*) mit dem Wert des bisherigen Verhaltenszustandes (*state*) initialisiert.

Dann startet der eigentliche Algorithmus mit der Bestimmung des Ereignis-Zustandes (*eventState*) des aktuellen Ereignis-Objektes, es wird die *happens*-Methode des Ereignis-Objektes aufgerufen. Danach wird der Sequenz-Zustand in Abhängigkeit vom Ereignis-Zustand aktualisiert.

Wenn alle Sequenz-Bedingungen abgearbeitet wurden, d.h. das aktuelle Ereignis-Objekt das letzte in der Liste *sequenceEvents* ist, und der Ereignis-Zustand den Wert *end possible* bzw. *finished* hat, dann wechselt der Sequenz-Zustand entsprechend.

Falls noch nicht das letzte Ereignis-Objekt bei der Untersuchung erreicht ist, und der Ereignis-Zustand den Wert *finished* hat, dann wird das folgende Ereignis-Objekt der *sequenceEvents* zum aktuellen Ereignis-Objekt und der eigentliche Algorithmus wird wiederholt.

Die Bestimmung des Zustandes der Immer-Bedingungen, also der Ablauf der Methode *happensAlwaysEvents* ist wesentlich einfacher. Jedes dieser Ereignis-Objekte wird mit dessen *happens*-Methode ausgewertet und deren Zustände werden nacheinander kombiniert. Das Ergebnis einer solchen Kombination ist wiederum durch die Matrix in Tabelle 4.4 auf Seite 96 gegeben. Da die durch die Matrix dargestellte Kombinations-Operation assiozativ ist, spielt die Reihenfolge der Kombinationen keine Rolle.

4.5.6.2 Anwendung des Algorithmus zur Verhaltens-Erkennung

Durch den Aufruf der Methode *happens* wird der Zustand eines Verhaltens-Objektes entsprechend der aktuellen Beobachtung bestimmt und die Belegung der Attribute des Verhaltens-Objektes wird aktualisiert. Die Anwendung der Methode *happens* für die Verhaltens-Erkennung ist in Algorithmus 4.3 dargestellt.

Algorithmus 4.3: Anwendung der Verhaltens-Erkennung

```
//aktuelle Verhaltens-Objekte initialisieren
currentBehaviors.addNewBehaviors();
...
while( observation.next() == YES ) { //solange Beobachtungen da
  //jedes aktuelle Verhaltens-Objekt mit Beobachtung untersuchen
  forall behavior in currentBehaviors {
    state = behavior.happens();
    if( state == finished || state == canceled ) {
      //Nutzung des erkannten Verhaltens
      ...
      //wenn Verhalten beendet dann nicht weiter untersuchen
      currentBehaviors.remove( behavior );
    }
  }
  //bei Bedarf Aufnahme neuer Verhaltens-Objekte
  currentBehaviors.addNewBehaviors();
}
```

Um ein Verhalten wahrzunehmen, muß ein Verhaltens-Objekt solange mit neuen Beobachtungen untersucht werden, bis sich dieses Verhaltens-Objekt entweder im Zustand *finished* oder im Zustand *canceled* befindet. Wenn einer dieser Zustände erreicht ist, kann das erkannte Verhalten auf unterschiedliche Art und Weise weiterverwendet werden. In dieser Arbeit wird es als Lösung eines Auslöser-Verhaltens-Falles genutzt (vgl. Abschnitt 3.5 auf Seite 37 oder 5.4.1 auf Seite 114).

Damit muß zur Erkennung der Verhalten aller Verhaltens-Klassen immer je ein Verhaltens-Objekt pro Klasse zur Untersuchung mit den Beobachtungen zur Verfügung stehen. Die gerade untersuchten Verhaltens-Objekte werden im Algorithmus 4.3 mit der Verhaltens-Objekte-Menge *currentBehaviors* bezeichnet. Neue Verhaltens-Objekte werden dieser Menge mit der Methode *addNewBehaviors* hinzugefügt.

Kapitel 5

Fallbasierte Generierung von Verhaltens-Modellen

In diesem Kapitel wird die Generierung von teamspezifischen Verhaltens-Modellen für Interaktionspartner behandelt. Das verwendete Verhaltens-Modell ist fallbasiert, enthält also Fälle die das Verhalten der Interaktionspartner unter bestimmten Bedingungen beschreiben. Die Beschreibung von Verhalten erfolgt dabei durch dessen charakteristische Werte. Für die automatische Generierung von Fällen, auf Basis früherer Beobachtungen der IP, wird für jedes beobachtete Verhalten dessen Ursache in Form eines Auslösers bestimmt. Ein Auslöser stellt die Problembeschreibung und das Verhalten die Lösung eines Falles dar.

Es wird sowohl ein Ähnlichkeitsmaß zwischen Auslösern als auch zwischen Verhalten verwendet. Das Ähnlichkeitsmaß für Auslöser dient zur Verhaltens-Vorhersage in Situationen, für die keine entsprechenden Auslöser-Verhaltens-Fälle existieren. Das Ähnlichkeitsmaß zwischen Verhalten wird zur Evaluierung der Verhaltens-Vorhersage und zur automatischen Anpassung des Auslöser-Ähnlichkeitsmaßes verwendet.

Das Verhaltens-Modell eines modellierten Teams wird generiert, indem es in Spielen gegen andere Teams beobachtet wird und Auslöser-Verhaltens-Fälle des modellierten Teams erzeugt werden. Dieses Verhaltens-Modell kann dann in weiteren Spielen des modellierten Teams zur Vorhersage des Verhaltens der Spieler dieses Teams verwendet werden.

Zuerst werden die verwendeten Auslöser-Muster für die Verhaltens-Muster Paß, Torschuß, Klären und Dribbeln vorgestellt (Abschnitt 5.1). Danach wird der Prozess der Bestimmung von Auslösern formalisiert (Abschnitt 5.2). Die charakteristischen Attribute der Verhaltens-Muster werden herausgearbeitet und durch Verhaltens-Essenz-Muster beschrieben (Abschnitt 5.3). Auf der Basis von Auslösern und Verhaltens-Essenzen werden die Auslöser-Verhaltens-Fälle eingeführt und es wird geklärt, wie diese generiert und in eine Fallbasis integriert werden (Abschnitt 5.4). Die Ähnlichkeitsmaße zwischen den Auslösern sowie zwischen den Verhalten werden spezifiziert (Abschnitt 5.5). Schließlich wird eine objektorientierte Realisierung für die Generierung von Verhaltens-Modellen angegeben (Abschnitt 5.6).

Die Anwendung von Verhaltens-Modellen wird in Kapitel 6 betrachtet.

5.1 Die Auslöser für die Verhaltens-Muster

Mit einem Auslöser wird der relevante Teil einer Entscheidungs-Situation für das Zustandekommen eines Verhaltens beschrieben (vgl. Definition 3.7 auf Seite 36). In diesem Abschnitt ist zu klären, was als relevanter Teil einer Entscheidungs-Situation zu verstehen ist. Dazu werden einige Vorbetrachtungen für die Wahl von Auslöser-Mustern durchgeführt (Abschnitt 5.1.1). Im Anschluß werden die verwendeten Auslöser-Muster für die Verhaltens-Muster Paß, Torschuß, Klären und Dribbeln vorgestellt (Abschnitte 5.1.2 bis 5.1.4).

5.1.1 Vorbemerkungen zur Wahl von Auslöser-Mustern

Die Auslöser-Muster werden mit der Absicht spezifiziert, die Grundlagen für das Zustandekommen eines Verhaltens durch die IP so abzubilden, daß auf deren Basis das Verhalten der IP vorhergesagt werden kann. Für ein Auslöser-Muster müssen also relevante Attribute für das Zustandekommen von Verhalten eines Verhaltens-Musters bestimmt werden. Als Ausgangsbasis stehen sämtliche Attribute einer Situation zur Verfügung, wie z.B. die Position des Balles. Bei der Spezifikation von Auslöser-Mustern sind die folgenden Anforderungen zu beachten:

- ein Auslöser muß die charakteristischen Daten (relevant für das Zustandekommen eines Verhaltens) einer Entscheidungs-Situation umfassen
- der Vergleich zweier Auslöser auf Ähnlichkeit muß sehr schnell erfolgen, so daß ein Auslöser auch noch mit mehreren anderen Auslösern in Echtzeit verglichen werden kann
- Auslöser müssen komplett aus der Entscheidungs-Situationen ableitbar sein, damit mögliche Auslöser für beliebige Situationen bestimmbar sind – insbesondere dürfen Daten des dem Auslöser zugeordneten Verhaltens nicht verwendet werden

Mit der ersten Anforderung wird die Vollständigkeit von Auslösern thematisiert. Welche Daten einer Situation die IP für ihre Entscheidungsfindung als relevant betrachten ist dem MA verschlossen, da er die Interna der IP nicht kennt. Verschiedene IP können auch verschiedene Daten als relevant betrachten. Bei der Wahl der Auslöser-Muster durch den Designer gilt es demnach, die wahrscheinlich relevanten Attribute zu ermitteln.

Mit der Anzahl der im Auslöser-Muster verwendeten Attribute steigt die Zeit zum Vergleich zweier Auslöser auf Ähnlichkeit. Es ist also ein Kompromiß zwischen der Vollständigkeit von Auslösern und der Minimierung der Vergleichs-Zeit zweier Auslöser zu finden. Wie im folgenden Absatz erörtert wird, ist eine Verwendung der kompletten Situation als Auslöser nicht ratsam.

Um ähnliche Auslöser zu erkennen, die durch den Positionstausch zweier Spieler einer Mannschaft entstehen, muß prinzipiell jede Spielerposition des einen Auslösers mit allen Spielerpositionen des anderen Auslösers verglichen werden, sofern die Spieler zum selben Team gehören. Der Vergleich zweier

Auslöser würde demnach aus mindestens $2 * 11^2 + 1$ Positionsvergleichen bestehen. Ein Vergleich eines Auslösers mit mehreren anderen Auslösern ist damit in Echtzeit nur schwer realisierbar. Auch eine Suche nach ähnlichen Auslösern im Zustandsraum hat kaum Aussicht auf Erfolg, da dieser sehr groß ist. Bei einer Begrenzung auf die Positionen der Spieler und des Balles und bei einer Diskretisierung der Positionen auf ganze Meter ergibt sich ein Zustandsraum von über 7000^{23} möglichen Positionen (für jeden der 22 Spieler und den Ball ergeben sich über 7000 verschiedene Positionen auf dem Spielfeld).

Damit ein Auslöser mit mehreren Auslösern in Echtzeit verglichen werden kann, begrenzen wir uns bei der Spezifikation der Auslöser-Muster auf die notwendigsten Attribute. Im Mittelpunkt aller betrachteten Verhaltens-Muster steht der Transfer des Balles zu einem Spieler oder einem Ort. Dieser Ziel-Spieler oder Ziel-Ort ist neben der Position des ballführenden Spielers ein wichtiges Attribut aller in den nächsten Abschnitten spezifizierten Auslöser-Muster.

Die Attribute werden in **primäre** und **sekundäre** Attribute unterteilt. Die primären Attribute werden bei der Suche nach passenden Auslösern und damit bei der Ermittlung der Ähnlichkeit zweier Auslöser benutzt. Die sekundären Attribute stehen für darüber hinausgehende Betrachtungen zur Verfügung.

5.1.2 Das Auslöser-Muster für das Paß-Verhalten

Die beiden zentralen Attribute für das Paß-Auslöser-Muster sind die Positionen von Kicker und Empfänger. Diese und andere wichtige Attribute des Auslöser-Musters sind in Abbildung 5.1 graphisch dargestellt.

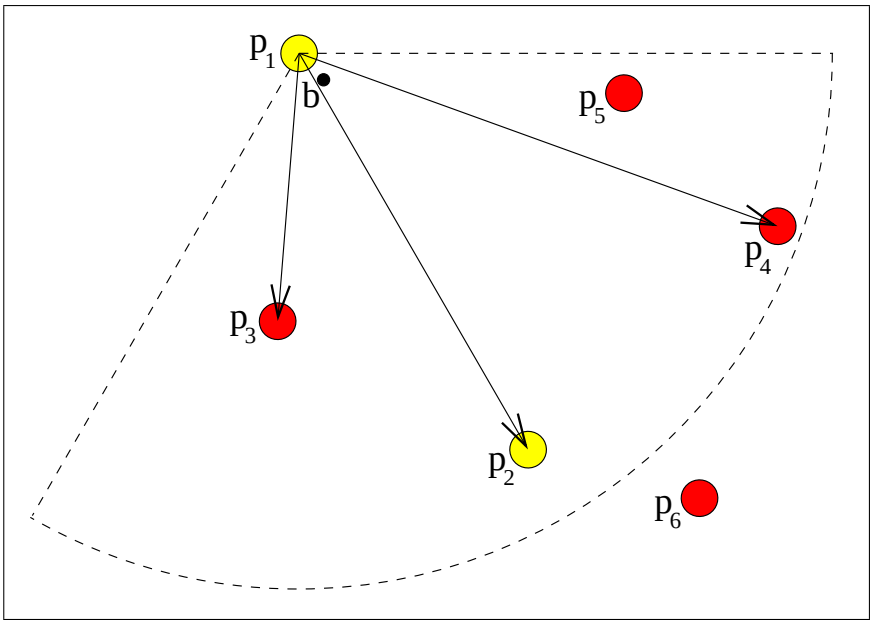


Abbildung 5.1: Wesentliche Bestandteile des Auslösers eines Passes von Spieler p_1 zu Spieler p_2

Die Position des Kickers p_1 ist ein wichtiges Attribut. Es wird benötigt, um die Abhängigkeit des Paß-Verhaltens von der Position des Kickers abzubilden.

Ein weiteres wichtiges Attribut ist die relative Position des Empfängers p_2 zum Kicker p_1, also der Vektor $\overrightarrow{p_1p_2}$. So wird ein Kicker wahrscheinlich nicht zu einem potentiellen Empfänger passen, wenn sich dieser in unmittelbarer Nähe oder sehr weit weg vom Kicker befindet. Im Zusammenhang mit der Position des Kickers ist auch die Richtung zum Empfänger relevant, so wird der Kicker keinen Rückwärtspaß ausführen, wenn die Möglichkeit zu einem Torschuß besteht.

Die Spieler der gegnerischen Mannschaft treten bei der Durchführung von Pässen als Hindernisse auf. Dabei sind die gegnerischen Spieler relevant, die sich in einem bestimmten Bereich zwischen Kicker und Empfänger befinden. Auch Gegner kurz hinter dem Empfänger können den Paß stören. Der Bereich, in dem Gegner als Hindernisse bei der Durchführung des Passes angesehen werden, ist durch ein Kreissegment[1] gegeben, welches in der Abbildung 5.1 mit einer Strichellinie dargestellt ist.

Als relevante Gegenspieler werden der erste Spieler zur rechten Seite und der erste Spieler zur linken Seite des Vektors $\overrightarrow{p_1p_2}$ betrachtet, die sich innerhalb des Kreissegments befinden. Als erster Spieler wird jeweils der Spieler p_i ausgewählt, dessen Winkel zwischen den Vektoren $\overrightarrow{p_1p_2}$ und $\overrightarrow{p_1p_i}$ minimal ist. In Abbildung 5.1 ist damit p_3 der erste Spieler zur rechten Seite und p_4 der erste Spieler zur linken Seite. Andere Gegenspieler werden für die Ausführung als nicht relevant betrachtet, so daß die Spieler p_5 (Einfluß bereits durch Spieler p_4 abgedeckt) und p_6 (liegt außerhalb des Kreissegments) als unbedeutend für den Paß-Auslöser angesehen werden.

Als Attribute werden die Winkel (zwischen den Vektoren $\overrightarrow{p_1p_2}$ und $\overrightarrow{p_1p_i}$) und die Entfernungen ($\overrightarrow{p_1p_i}.length$) zum jeweils ersten Gegner verwendet. Damit ergeben sich sechs primäre Attribute für das Paß-Auslöser-Muster.

Primäre Attribute

- Position des Kickers p_1
- Vektor $\overrightarrow{p_1p_2}$ vom Kicker p_1 zum Empfänger p_2
- Winkel $\measuredangle(\overrightarrow{p_1p_3}, \overrightarrow{p_1p_2})$ zum ersten Gegner p_3 rechts
- Entfernung $\overrightarrow{p_1p_3}.length$ zum ersten Gegner p_3 rechts
- Winkel $\measuredangle(\overrightarrow{p_1p_4}, \overrightarrow{p_1p_2})$ zum ersten Gegner p_4 links
- Entfernung $\overrightarrow{p_1p_4}.length$ zum ersten Gegner p_4 links

Unter der Vorraussetzung, daß ein Team öfter oder immer mit der gleichen Aufstellung spielt, kann es sein, daß ein Kicker immer spezielle Spieler als Empfänger von Pässen bevorzugt. Mit der Aufnahme der Spielernummern von Kicker und Empfänger als sekundäre Attribute kann diese eventuelle Abhängigkeit abgebildet werden.

Der Zeitpunkt und der Spielzustand[2] der Entscheidungs-Situation werden als weitere sekundäre Attribute aufgenommen. Bei sonst gleichen Werten eines Auslösers ist bei verschiedenen Spielzuständen unterschiedliches Verhalten seitens der IP zu beobachten.

[1] Es wird ein Kreissegment verwendet, dessen Zentrum sich in der Position des Kickers befindet, dessen Radius 3 Meter größer ist als die Entfernung vom Kicker zum Empfänger und dessen Öffnungswinkel zum Vektor Kicker-Empfänger jeweils 60 Grad beträgt.

[2] siehe Abschnitt 3.2 auf Seite 32

Da davon auszugehen ist, daß sich zumindest einige Teams an die Eigenschaften des anderen (gegnerischen) Teams anpassen, ist es möglich, daß das von den modellierten IP ausgeführte Verhalten auch vom Verhalten des anderen Teams abhängt. Deshalb wird der Name des anderen Teams als sekundäres Attribut aufgenommen. Zusammengefasst ergeben sich fünf sekundäre Attribute.

Sekundäre Attribute
- Spielernummer vom Kicker p_1
- Spielernummer vom Empfänger p_2
- Zeitpunkt
- Spielzustand
- Name des anderen Teams

5.1.3 Das Auslöser-Muster für das Torschuß-Verhalten

Das Auslöser-Muster Torschuß ähnelt dem Paß-Auslöser-Muster. An Stelle des Empfängers als Ziel des Balles tritt das Zentrum des gegnerischen Tores. Abbildung 5.2 zeigt die wichtigsten Attribute des Auslöser-Musters Torschuß. Da die Position des gegnerischen Tor-Zentrums eindeutig festgelegt ist, wird entweder nur die Position des Kickers oder der Vektor zum Tor als Attribut benötigt. Hier wird der Vektor vom Kicker zum gegnerischen Tor verwendet. Dieser Vektor wird im folgenden mit $\overrightarrow{p_1 g_c}$ bezeichnet.

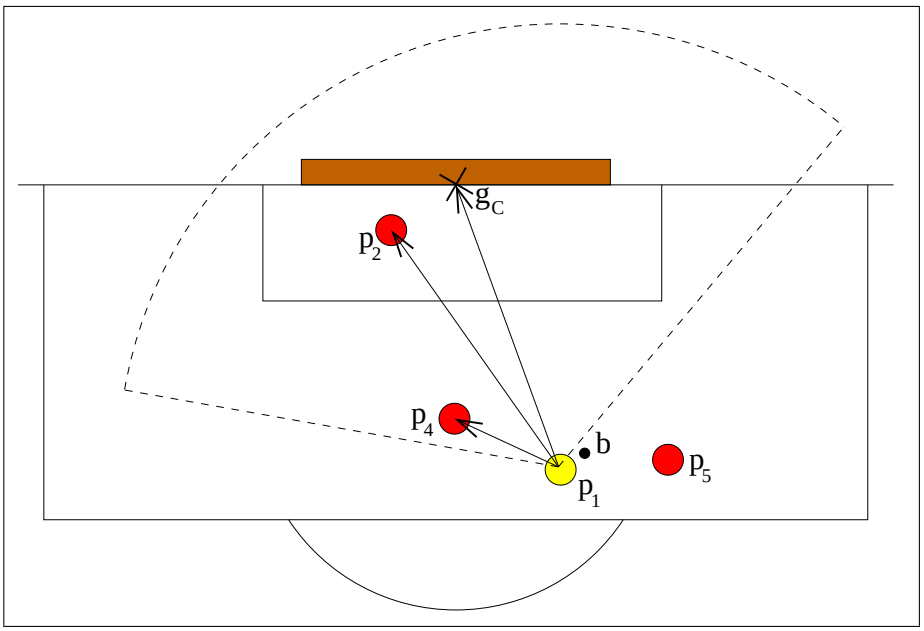


Abbildung 5.2: Wesentliche Bestandteile des Auslösers eines Torschusses

Gegnerische Spieler werden wiederum als relevant für das Auslöser-Muster betrachtet, wenn sie sich in einem bestimmten Bereich zwischen Kicker und Tor oder kurz hinter dem Tor befinden. Es wird wiederum ein Kreissegment verwendet, welches durch eine Strichellinie in Abbildung 5.2 dargestellt ist. Der Radius des Kreissegment ist dabei größer gewählt als die Entfernung zum Tor-

Zentrum, da auch Gegner die sich hinter dem Tor befinden, einen Einfluß auf den Erfolg des Torschuß-Verhaltens haben können.

Der gegnerische Torwart zeichnet sich als spezieller Gegenspieler mit erweiterten Fähigkeiten aus. Aus diesem Grunde wird er als relevanter Spieler betrachtet, unabhängig auf welchen Positionen sich die anderen Gegner befinden. Als weitere relevante Gegenspieler werden der erste Spieler rechts und der erste Spieler links vom Vektor $\overrightarrow{p_1 g_c}$ angesehen. Die Bestimmung der jeweils ersten Spieler wird genauso wie beim Auslöser-Muster Paß gehandhabt, mit dem einzigen Unterschied, daß der gegnerische Torwart dabei nicht berücksichtigt wird. In Abbildung 5.2 ist der Spieler p_2 der gegnerische Torwart und der Spieler p_4 ist der erste Gegner links vom Vektor $\overrightarrow{p_1 g_c}$. Rechts von diesem Vektor befindet sich kein Spieler im zu betrachtenden Kreissegment.

Als Attribute werden die Winkel und die Entfernungen zum Torwart und zu den jeweils ersten Gegnern verwendet. Damit ergeben sich sieben primäre Attribute für das Auslöser-Muster Torschuß. Die sekundären Attribute sind nahezu identisch mit den sekundären Attributen des Verhaltens-Musters Paß.

Primäre Attribute
- Vektor $\overrightarrow{p_1 g_c}$ vom Kicker p_1 zum gegnerischen Torzentrum
- Winkel $\measuredangle(\overrightarrow{p_1 p_2}, \overrightarrow{p_1 g_c})$ zum gegnerischen Torwart p_2
- Entfernung $\overrightarrow{p_1 p_2}.length$ zum gegnerischen Torwart p_2
- Winkel $\measuredangle(\overrightarrow{p_1 p_3}, \overrightarrow{p_1 g_c})$ zum ersten Gegner p_3 rechts
- Entfernung $\overrightarrow{p_1 p_3}.length$ zum ersten Gegner p_3 rechts
- Winkel $\measuredangle(\overrightarrow{p_1 p_4}, \overrightarrow{p_1 g_c})$ zum ersten Gegner p_4 links
- Entfernung $\overrightarrow{p_1 p_4}.length$ zum ersten Gegner p_4 links

Sekundäre Attribute
- Spielernummer vom Kicker p_1
- Zeitpunkt
- Spielzustand
- Name des anderen Teams

5.1.4 Das Auslöser-Muster für das Dribbel-Verhalten

Bei der Spezifikation des Dribbel-Auslöser-Musters tritt das Problem auf, daß kein potentielles Ziel wie beim Paß oder beim Torschuß vorhanden ist, welches man als Ausgangsbasis für die Spezifikation verwenden kann. Nur durch Betrachtung des ausgeführten Verhaltens kann eine Richtung als Ziel identifiziert werden. Damit ist es schwierig, basierend auf Entscheidungs-Situationen die relevanten Attribute für dieses Auslöser-Muster zu bestimmen.

Um dieses Problem zu umgehen werden virtuelle Ziele eingeführt. Solche virtuellen Ziele sind fixierte Richtungen α auf dem Spielfeld. Als fixierte Richtungen werden Richtungen alle 60 Grad betrachtet[3], wobei die Richtung parallel zum Seitenaus (0 Grad), eine davon ist. Insgesamt gibt es 6 derartige Richtungen. Für ein beobachtetes Dribbel-Verhalten wird als virtuelles Ziel

[3]Die Entscheidung für Dribbelrichtungen alle 60 Grad orientiert sich an der Größe des für das Paß-Auslöser-Muster verwendeten Kreissegments.

die Richtung verwendet, die der Dribbel-Richtung am nächsten ist. Für eine Entscheidungs-Situation wird für jede der 6 Richtungen ein Dribbel-Auslöser bestimmt.

Als relevante gegnerische Spieler werden die ersten Gegner nach rechts (p_3) und links (p_4) von der virtuellen Zielrichtung α angesehen. Die Bestimmung dieser Gegner erfolgt wie bei den vorherigen Auslöser-Mustern. Der relevante Bereich ist wiederum durch ein Kreissegment gegeben. Da dieses Auslöser-Muster auch für einen Teil der Klären-Verhalten verwendet wird (siehe nächster Abschnitt), wird der Radius des Kreissegments in Abhängigkeit von der Situation bestimmt[4].

Das Dribbel-Verhalten kann auch davon abhängen, ob der ballführende Spieler p_1 von Gegnern bedrängt wird. Deshalb wird die Entfernung des dem Kicker am nächsten stehenden Gegners p_2 mit als Attribut aufgenommen. Von den ersten Gegnern nach rechts und nach links werden deren relative Winkel zur virtuellen Zielrichtung und deren Entfernungen als primäre Attribute verwendet. Die sekundären Attribute sind die selben wie beim Auslöser-Muster Torschuß.

Primäre Attribute
- Position des Kickers p_1
- virtuelle Zielrichtung α
- Entfernung $\overrightarrow{p_1p_2}.length$ des dem Kicker am nächsten stehenden Gegners p_2
- Winkel $|\overrightarrow{p_1p_3}.angle - \alpha|$ zum ersten Gegner p_3 rechts
- Entfernung $\overrightarrow{p_1p_3}.length$ zum ersten Gegner p_3 rechts
- Winkel $|\overrightarrow{p_1p_4}.angle - \alpha|$ zum ersten Gegner p_4 links
- Entfernung $\overrightarrow{p_1p_4}.length$ zum ersten Gegner p_4 links

Sekundäre Attribute
- Spielernummer vom Kicker p_1
- Zeitpunkt
- Spielzustand
- Name des anderen Teams

5.1.5 Das Auslöser-Muster für das Klären-Verhalten

Bei der Spezifikation des Verhaltens-Musters Klären wurden die beiden Ausprägungen „Vorlegen“ und „Mitspieler“ identifiziert (vgl. Abschnitt 4.4.3 auf Seite 70).

Bei der Ausprägung „Vorlegen“ erlangt der Kicker selber wieder die Kontrolle über den Ball. Diese Ausprägung unterscheidet sich also nur in der Ballgeschwindigkeit vom Verhaltens-Muster Dribbeln und kann auch als schnelles Dribbeln interpretiert werden. Für die Modell-Generierung und die Vorhersage von Verhalten wird die Ausprägung „Vorlegen“ als Dribbel-Verhalten betrachtet und damit das Auslöser-Muster Dribbeln verwendet.

Die Ausprägung „Mitspieler“ des Verhaltens-Musters Klären ist dadurch gekennzeichnet, daß ein Mitspieler die Kontrolle über den Ball erlangt. Diese

[4] Befinden sich Mitspieler innerhalb eines Winkels von 30 Grad um die virtuelle Zielrichtung wird der Radius des Kreissegments auf die Entfernung zum Mitspieler beschränkt, andernfalls wird der Radius auf 40 Meter festgelegt.

Ausprägung kann damit auch als ein „Paß in den Raum“ interpretiert werden. Wir betrachten sie daher von jetzt an als Paß-Verhalten und verwenden damit das Auslöser-Muster vom Paß.

Bei einem fehlerhaften Klären-Verhalten erlangt entweder ein gegnerischer Spieler die Ballkontrolle oder der Spielzustand wechselt. Solche Ausprägungen können nur als fehlgeschlagenes Dribbel-Verhalten interpretiert werden, da ein potentieller Zielspieler nicht ermittelt werden kann.

5.2 Formalisierung der Auslöser-Bestimmung

Der Prozess der Auslöser-Bestimmung setzt sich zusammen aus:

- der Spezifikation von Auslösern (engl. specify trigger) und
- der Erkennung von Auslösern (engl. recognize trigger).

In Abbildung 5.3 sind diese beiden Teilprozesse dargestellt. Wie bei der Verhaltens-Bestimmung wird die Spezifikation von Auslösern durch einen Designer vorgenommen, der für jedes Verhaltens-Muster ein Auslöser-Muster definiert. Diese Auslöser-Muster werden sowohl genutzt, um den Auslöser eines Verhaltens, als auch die möglichen Auslöser einer Entscheidungs-Situation zu bestimmen. In diesem Abschnitt wird die Spezifikation von Auslösern (Abschnitt

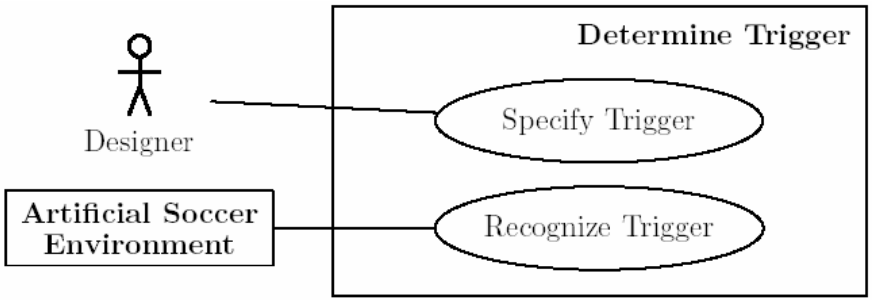


Abbildung 5.3: Struktur der Auslöser-Bestimmung

5.2.1) und die Erkennung von Auslösern (Abschnitt 5.2.2) formalisiert.

5.2.1 Spezifikation von Auslösern

Da jedes Auslöser-Muster einem Verhaltens-Muster zugeordnet ist, können die Namen der entsprechenden Verhaltens-Muster verwendet werden. Damit besteht jeder Auslöser aus:

- einem Namen $name \in \mathbb{NB}$,[5]
- einer Sequenz von Primärwerten und
- einer Sequenz von Sekundärwerten.

[5] $\mathbb{NB}$ wurde als Menge der Namen aller Verhalten eingeführt (Abschnitt 4.3.3 auf Seite 58).

Es werden also zwei verschiedene Arten von Werte-Sequenzen unterschieden. Im Vorfeld der Formalisierung der Auslöser und Auslöser-Muster wird die Menge der Sequenztypen eingeführt.

$\mathbb{ST}$ ist die Menge der Sequenztypen. Sie besteht aus den zwei Elementen 'primary' und 'secondary', d.h. $\mathbb{ST} = \{\text{primary}, \text{secondary}\}$.

Bei der Spezifikation der Auslöser-Muster wurden verschiedene primäre und sekundäre Attribute verwendet, die in Tabelle 5.1 zusammengefasst sind. In der linken Spalte sind die Namen der Attribute angegeben, wobei die Vektoren vom Kicker zum Empfänger bzw. zum Torzentrum in der Tabelle mit „relative Positionen" bezeichnet werden. Die mittlere Spalte gibt die verwendete Bezeichnung der Menge an und die rechte Spalte macht Angaben über die Elemente der jeweiligen Menge.

Attribut	Mengenname	Erläuterungen
Kickerpositionen	$\mathbb{KP}$	Teilmenge von $\mathbb{V}$
relative Positionen	$\mathbb{RP}$	Teilmenge von $\mathbb{V}$
Winkel zum Gegner	$\mathbb{OA}$	Teilmenge von $\mathbb{R}$
Entfernungen zum Gegner	$\mathbb{OD}$	Teilmenge von $\mathbb{R}$
virtuelle Zielrichtungen	$\mathbb{VD}$	$\{0, 60, 120, 180, 240, 300\}$
Spielernummern	$\mathbb{NR}$	$\{1, \ldots, 11\}$
Zeitpunkte	$\mathbb{T}$	bereits definiert
Spielzustände	$\mathbb{GS}$	{'im Spiel', 'Freistoß etc.'}
Gegnernamen	$\mathbb{ON}$	beliebige Zeichenketten

Tabelle 5.1: Die für Auslöser verwendeten Attribute und ihre Mengen

Da Auslöser mit den selben Namen durch ein und dasselbe Auslöser-Muster repräsentiert werden sollen, werden zuerst die Mengen aller Werte-Sequenzen mit dem Namen $name \in \mathbb{NB}$ und dem Sequenztyp $type \in \mathbb{ST}$ eingeführt. Anschließend werden die Mengen der Auslöser und der Auslöser-Muster definiert.

$\texttt{triggerTupel}_{type}^{name}$ bezeichnet die Menge aller **Werte-Sequenzen von Auslösern** mit dem Namen $name \in \mathbb{NB}$ und dem Sequenztyp $type \in \mathbb{ST}$. Jede Werte-Sequenz-Menge $\texttt{triggerTupel}_{type}^{name}$ entspricht dabei einer Sequenz von Attributen, es gilt

$$\texttt{triggerTupel}_{type}^{name} = \mathbb{A}_1 \times \cdots \times \mathbb{A}_n \tag{5.1}$$

wobei jedes $\mathbb{A}_i$ mit $1 \leq i \leq n$ eine der Attributmengen aus Tabelle 5.1 bezeichnet. Die Anzahl der Attributmengen n und die Angabe, um welche Attributmengen es sich jeweils bei den Mengen $\mathbb{A}_i$ handelt, ist durch den Namen *name* und den Sequenztyp *type* eindeutig definiert.

Beispielsweise ist $\texttt{triggerTupel}_{\text{primary}}^{\text{pass}} = \mathbb{KP} \times \mathbb{RP} \times \mathbb{OA} \times \mathbb{OD} \times \mathbb{OA} \times \mathbb{OD}$ die Menge der Werte-Sequenzen mit den Namen pass und dem Sequenztyp primary.

`triggers` bezeichnet die Menge aller **Auslöser**. Ein Auslöser besteht aus einem Namen, einer Sequenz von Primärwerten und einer Sequenz von Sekundärwerten. Der Name ist ein Element der Menge $\mathbb{NB}$ und die Sequenzen sind Elemente der passenden Werte-Sequenzen $\texttt{triggerTupel}^{name}_{type}$.

$$\begin{aligned}\texttt{triggers} = \big\{ & name\,(pSeq, sSeq) : name \in \mathbb{NB}, \\ & pSeq \in \texttt{triggerTupel}^{name}_{\text{primary}},\ sSeq \in \texttt{triggerTupel}^{name}_{\text{secondary}} \big\}\end{aligned} \tag{5.2}$$

`triggerPatterns` bezeichnet die Menge aller **Auslöser-Muster**. Ein Auslöser-Muster $TP = name\,(\texttt{triggerTupel}^{name}_{\text{primary}}, \texttt{triggerTupel}^{name}_{\text{secondary}})$ mit $name \in \mathbb{NB}$ repräsentiert alle Auslöser gleichen Namens. Damit ergibt sich die Menge aller Auslöser-Muster als

$$\begin{aligned}\texttt{triggerPatterns} = \big\{ & name\,(\texttt{triggerTupel}^{name}_{\text{primary}}, \\ & \texttt{triggerTupel}^{name}_{\text{secondary}}) : name \in \mathbb{NB} \big\}.\end{aligned} \tag{5.3}$$

Da jede Werte-Sequenz-Menge $\texttt{triggerTupel}^{name}_{type}$ gleichzeitig als eine Sequenz von Attributen angesehen werden kann, verwenden wir für die Auslöser-Muster auch Attributsbezeichnungen und benennen Auslöser-Muster auch mit $TP = name\,(\langle ap_1, \ldots, ap_k\rangle, \langle as_1, \ldots, as_l\rangle)$, wobei die ap_i und as_j jeweils beliebige Elemente der entsprechenden Attributmengen $\mathbb{A}_i$ bzw. $\mathbb{A}_j$ bezeichnen.

Bei Verwendung der Attributsbezeichnungen ergibt sich für das Auslöser-Muster Paß $TP_{\text{pass}} = \text{pass}\,(\langle ap_1, \ldots ap_6\rangle, \langle as_1 \ldots as_5\rangle)$, wobei $ap_1 \in \mathbb{KP}$, $ap_2 \in \mathbb{RP}$, $ap_3, ap_5 \in \mathbb{OA}$, $ap_4, ap_6 \in \mathbb{OD}$, $as_1, as_2 \in \mathbb{NR}$, $as_3 \in \mathbb{T}$, $as_4 \in \mathbb{GS}$ und $as_5 \in \mathbb{ON}$ gilt.

Die Mengen für Auslöser und Auslöser-Muster wurden aufbauend auf den Mengen der Werte-Sequenzen von Auslösern definiert. Jedes Auslöser-Muster $TP = name\,(\texttt{triggerTupel}^{name}_{\text{primary}}, \texttt{triggerTupel}^{name}_{\text{secondary}})$ stellt eine Auslösermenge $\{T_1, \ldots T_m\}$ mit $T_i = name\,(pSeq_i, sSeq_i)$, $pSeq_i \in \texttt{triggerTupel}^{name}_{\text{primary}}$ und $sSeq_i \in \texttt{triggerTupel}^{name}_{\text{secondary}}$ dar. Jedes dieser Ereignisse T_i wird durch das Ereignis-Muster TP repräsentiert. Diese Beziehung wird im folgenden mit $T_i \Subset TP$ gekennzeichnet.

5.2.2 Erkennung von Auslösern

Für Entscheidungs-Situationen müssen alle potentiellen Auslöser ermittelt werden. Diese Aufgabe wird durch die Funktion $\text{identify}_{\text{S}}$ verrichtet. Bei erkannten Verhalten muß für dessen Entscheidungs-Situation der entsprechende Auslöser identifiziert werden. Dafür ist die Funktion $\text{identify}_{\text{T}}$ zuständig.

$\text{identify}_{\text{S}}$ ist die Funktion, die für Entscheidungs-Situationen alle Auslöser eines Auslöser-Musters bestimmt. Bei den betrachteten Auslöser-Mustern wird für jedes potentielle Ziel ein Auslöser ermittelt. Für den Paß wird also für jeden als Empfänger in Frage kommenden Mitspieler ein Auslöser ermittelt. Bei weit entfernten Mitspielern kann auf eine Ermittlung des

Auslösers verzichtet werden. Für den Torschuß wird der Auslöser bzgl. des gegnerischen Torzentrums bestimmt, wenn sich dieser nicht zu weit vom Kicker entfernt befindet. Für das Dribbeln wird für jede der potentiellen Zielrichtungen ein Auslöser identifiziert.

$$\begin{aligned} \text{identify}_\text{S} &: \texttt{situations} \times \texttt{triggerPatterns} \rightarrow 2^{\texttt{triggers}} \\ \text{identify}_\text{S}\big(S, TP\big) &= \big\{T_1, \ldots, T_n\big\} \quad \text{mit } T_i \Subset TP, i = 1, \ldots, n \end{aligned} \tag{5.4}$$

Die Ermittlung von Auslösern erfolgt aber nur unter der Voraussetzung, daß S eine Entscheidungs-Situation bzgl. des Auslöser-Musters TP ist.

identify$_\text{T}$ ist die Funktion, mit der ein Auslöser entsprechend dem Ziel eines erkannten Verhaltens bestimmt werden kann. Beim Paß ist das Ziel der Empfänger des Paßes, beim Torschuß das Torzentrum und beim Dribbeln die der sechs Zielrichtungen, die der Bewegungsrichtung des Dribblers am nächsten ist.

$$\begin{aligned} \text{identify}_\text{T} &: \texttt{situations} \times \texttt{triggerPatterns} \times \texttt{behaviors} \rightarrow \texttt{triggers} \\ \text{identify}_\text{T}\big(S, TP, B\big) &= T \quad \text{mit } T \Subset TP \end{aligned} \tag{5.5}$$

Dabei ist S die Entscheidungs-Situation für das Zustandekommen des Verhaltens B und TP ist das Auslöser-Muster, welches den selben Namen wie das Verhalten B hat.

5.3 Die Essenzen der Verhaltens-Muster

Bisher wurde Verhalten durch eine Sequenz von Ereignissen definiert, wobei jedes Verhalten durch die Werte seiner Ereignisse eindeutig beschrieben ist. Für die Generierung und Anwendung von Verhaltens-Modellen ist es nicht sinnvoll, alle Werte der Ereignisse eines Verhaltens zu betrachten, sondern nur die charakteristischen Werte.

Für die Spezifikation von Verhaltens-Essenz-Mustern müssen daher die wesentlichen Attribute der Ereignis-Muster identifiziert und in Sequenzen zusammengefasst werden. Es wird dabei wieder zwischen primären und sekundären Attributen unterschieden. Damit besteht eine Verhaltens-Essenz genauso wie ein Auslöser aus:

- einem Namen $name \in \mathbb{NB}$,
- einer Sequenz von Primärwerten und
- einer Sequenz von Sekundärwerten.

Im folgenden werden die primären und sekundären Attribute der betrachteten Verhaltens-Essenz-Muster erarbeitet.

Da die Spielernummern von Kicker und Empfänger schon Bestandteil der Auslöser-Muster sind, müssen sie nicht mehr als Attribute von Verhaltens-Essenz-Muster aufgenommen werden. Die Startzeit des Verhaltens ist ebenso

schon Bestandteil der Auslöser-Muster. Statt der Endzeit wird die **Dauer des Verhaltens** als Attribut aufgenommen. Ob das Verhalten **erfolgreich oder erfolglos** war, wird auch mit einem Attribut festgehalten. Bei erfolglosen Verhalten wird die **Fehlerursache** als Attribut aufgenommen.

Die anderen Attribute sind jeweils vom entsprechenden Verhaltens-Muster abhängig. Im folgenden werden die Attribute der Verhaltens-Essenz-Muster Paß, Torschuß und Dribbeln aufgeführt (Abschnitte 5.3.1 bis 5.3.3). Das Verhaltens-Muster Klären wird wie schon erläutert (vgl. Abschnitt 5.1.5 auf Seite 107) auf die Verhaltens-Muster Paß und Dribbeln aufgeteilt. Danach werden einige Bemerkungen zur Formalisierung von Verhaltens-Essenzen und deren Muster gemacht (Abschnitt 5.3.4).

5.3.1 Die Essenz des Verhaltens-Musters Paß

Die wesentlichen Eigenschaften des Verhaltens-Musters Paß (vgl. Verhaltens-Muster 2 auf Seite 51) sind die **anfängliche Ballgeschwindigkeit** $ballSpeed \in \mathbb{V}$ und die **Bewegung des Empfängers** $playerMovement \in \mathbb{V}$. Von der Ballgeschwindigkeit werden dabei der Winkel und die Länge des Geschwindigkeitsvektors als eigene Attribute aufgenommen. Zusammen mit den weiter oben betrachteten Attributen ergeben sich fünf primäre und ein sekundäres Attribut (Fehlerursache).

Primäre und sekundäre Attribute

- Länge des Ballgeschwindigkeitsvektors $ballSpeed.length$
- Winkel des Ballgeschwindigkeitsvektors $ballSpeed.angle$
- Bewegung des Empfängers $playerMovement$
- Dauer des Verhaltens
- Erfolg oder Nichterfolg
- Fehlerursache

5.3.2 Die Essenz des Verhaltens-Musters Torschuß

Beim Verhaltens-Muster Torschuß (vgl. Verhaltens-Muster 5 auf Seite 70) treten ähnliche Attribute wie beim Verhaltens-Muster Paß auf, lediglich das Attribut der Empfänger-Bewegung existiert nicht. Das genaue Ziel des Torschußes $shotTarget \in \mathbb{V}$ muß nicht aufgenommen werden, da es durch die Position des Kickers (Attribut des Auslöser-Muster Torschuß) und durch die anfängliche Ballgeschwindigkeit definiert ist. Damit gibt es für dieses Verhaltens-Muster vier primäre Attribute und mit der Fehlerursache ein sekundäres Attribut.

Primäre und sekundäre Attribute

- Länge des Ballgeschwindigkeitsvektors $ballSpeed.length$
- Winkel des Ballgeschwindigkeitsvektors $ballSpeed.angle$
- Dauer des Verhaltens
- Erfolg oder Nichterfolg
- Fehlerursache

5.3.3 Die Essenz des Verhaltens-Musters Dribbeln

Beim Verhaltens-Muster Dribbeln (vgl. Verhaltens-Muster 3 auf Seite 68 und Verhaltens-Muster 4 auf Seite 69) gibt es im allgemeinen keine eindeutig identifizierbare anfängliche Ballgeschwindigkeit. Bei der Ausprägung des Dribbeln mit ständiger Ballkontrolle ist es nicht sinnvoll, die Ballgeschwindigkeit zu verwenden. Bei der Ausprägung mit teilweiser Aufgabe der Ballkontrolle ist es möglich, daß mehrere anfängliche Ballgeschwindigkeiten existieren. Es ergeben sich demnach drei primäre Attribute und ein sekundäres Attribut (Fehlerursache).

Primäre und sekundäre Attribute

- Bewegung des Dribblers $playerMovement$
- Dauer des Verhaltens
- Erfolg oder Nichterfolg
- Fehlerursache

5.3.4 Formalisierung der Spezifikation und der Identifikation von Verhaltens-Essenzen

Für die Formalisierung werden entsprechend der Mengen für Auslöser-Werte-Sequenzen, Auslöser und Auslöser-Muster die entsprechenden Mengen für Verhaltens-Essenzen definiert. Zur Vollständigkeit werden diese Mengen im folgenden aufgeführt. Die primären und sekundären Attribute der Muster für Verhaltens-Essenzen sind in Tabelle 5.2 zusammengefasst.

Attribut	Mengenname	Erläuterungen
Länge des Ballgeschwindigkeitsvektors	$\mathbb{SL}$	Teilmenge von $\mathbb{R}$
Winkel des Ballgeschwindigkeitsvektors	$\mathbb{SA}$	Teilmenge von $\mathbb{R}$
Bewegung eines Spielers	$\mathbb{PM}$	Teilmenge von $\mathbb{V}$
Verhaltens-Dauer	$\mathbb{DU}$	Teilmenge von $\mathbb{N}$
Erfolg oder Nichterfolg	$\mathbb{SU}$	$\{\text{ja}, \text{nein}\}$
Fehlerursache	$\mathbb{FR}$	Fehler-Ausprägungen aus Abschnitt 4.4.4.2 auf Seite 72

Tabelle 5.2: Die für Verhalten verwendeten Attribute und ihre Mengen

$\texttt{behavior}_E\texttt{Tupel}_{type}^{name}$ bezeichnet die Menge der **Werte-Sequenzen von Verhaltens-Essenzen** mit dem Namen $name \in \mathbb{NB}$ und dem Sequenztyp $type \in \mathbb{ST}$.

$$\texttt{behavior}_E\texttt{Tupel}_{type}^{name} = \mathbb{A}_1 \times \cdots \times \mathbb{A}_n \tag{5.6}$$

Dabei bezeichnet jedes der $\mathbb{A}_i$ einer der Attributmengen aus Tabelle 5.2.

$\texttt{behaviors}_E$ bezeichnet die Menge aller **Verhaltens-Essenzen**. Jede Verhaltens-Essenz besteht aus einem Namen $name \in \mathbb{NB}$ und zwei Werte-Sequenzen.

$$\begin{aligned}\texttt{behaviors}_E = \big\{ & name\,(pSeq, sSeq) : name \in \mathbb{NB}, \\ & pSeq \in \texttt{behavior}_E\texttt{Tupel}^{name}_{\text{primary}}, \\ & sSeq \in \texttt{behavior}_E\texttt{Tupel}^{name}_{\text{secondary}} \big\}\end{aligned} \tag{5.7}$$

$\texttt{behavior}_E\texttt{Patterns}$ bezeichnet die Menge aller **Muster von Verhaltens-Essenzen**. Ein Verhaltens-Essenz-Muster repräsentiert alle Verhaltens-Essenzen gleichen Namens.

$$\begin{aligned}\texttt{behavior}_E\texttt{Patterns} = \big\{ & name\,(\texttt{behavior}_E\texttt{Tupel}^{name}_{\text{primary}}, \\ & \texttt{behavior}_E\texttt{Tupel}^{name}_{\text{secondary}}) : name \in \mathbb{NB} \big\}\end{aligned} \tag{5.8}$$

Damit bestehen Auslöser und Verhaltens-Essenzen jeweils aus einem Namen, einer Sequenz von Primärwerten und einer Sequenz von Sekundärwerten.

Die Auswahl der entsprechenden Werte aus den erkannten Verhalten wird mit der Funktion identify_B durchgeführt.

identify_B ist die Funktion, mit der eine Verhaltens-Essenz basierend auf einem Verhalten bestimmt wird. Es werden die charakteristischen Werte des Verhaltens ausgewählt und eine Verhaltens-Essenz erzeugt. Die Verhaltens-Essenz hat dabei den gleichen Namen wie das Verhalten.

$$\begin{aligned}\text{identify}_\text{B} &: \texttt{behaviors} \rightarrow \texttt{behaviors}_E \\ \text{identify}_\text{B}\big(B\big) &= BE\end{aligned} \tag{5.9}$$

5.4 Fallbasis mit Auslöser-Verhaltens-Fällen

Mit den Auslösern und den Verhaltens-Essenzen stehen die Bestandteile von Auslöser-Verhaltens-Fällen zu Verfügung. Damit kann die Generierung von Fällen behandelt werden (Abschnitt 5.4.1). Desweiteren müssen neu generierte Fälle in eine bereits bestehende Fallbasis integriert werden (Abschnitt 5.4.2).

5.4.1 Generierung eines Falles

Bevor die Funktion zur Generierung eines Falles vorgestellt wird, führen wir die Menge aller Fälle ein. Im Anschluß wird ausgehend von einer Sequenz von Beobachtungen der komplette Prozess der Erzeugung eines Auslöser-Verhaltens-Falles durchgesprochen.

`cases` ist die Menge aller Fälle. Ein Fall besteht aus einem Auslöser und einer Verhaltens-Essenz, wobei Auslöser und Verhaltens-Essenz den gleichen Namen haben.

$$\begin{aligned}\texttt{cases} = \{(T, BE) : & T \in \texttt{triggers}, BE \in \texttt{behaviors}_E, \\ & T = name(\dots), BE = name(\dots)\}\end{aligned} \tag{5.10}$$

generateCase ist die Funktion, die aus einem Auslöser und einer Verhaltens-Essenz einen Auslöser-Verhaltens-Fall generiert.

$$\begin{aligned} \text{generateCase} &: \texttt{triggers} \times \texttt{behaviors}_E \rightarrow \texttt{cases} \\ \text{generateCase}\big(T, BE\big) &= \big(T, BE\big) \end{aligned} \tag{5.11}$$

Abbildung 5.4 zeigt beispielhaft die Generierung eines Auslöser-Verhaltens-Falles auf Basis einer Beobachtungssequenz. Im oberen Teil der Abbildung ist ein Interaktionspartner, im mittleren Teil die Multi-Agenten-Umgebung und im unteren Teil der Modellierende Agent dargestellt.

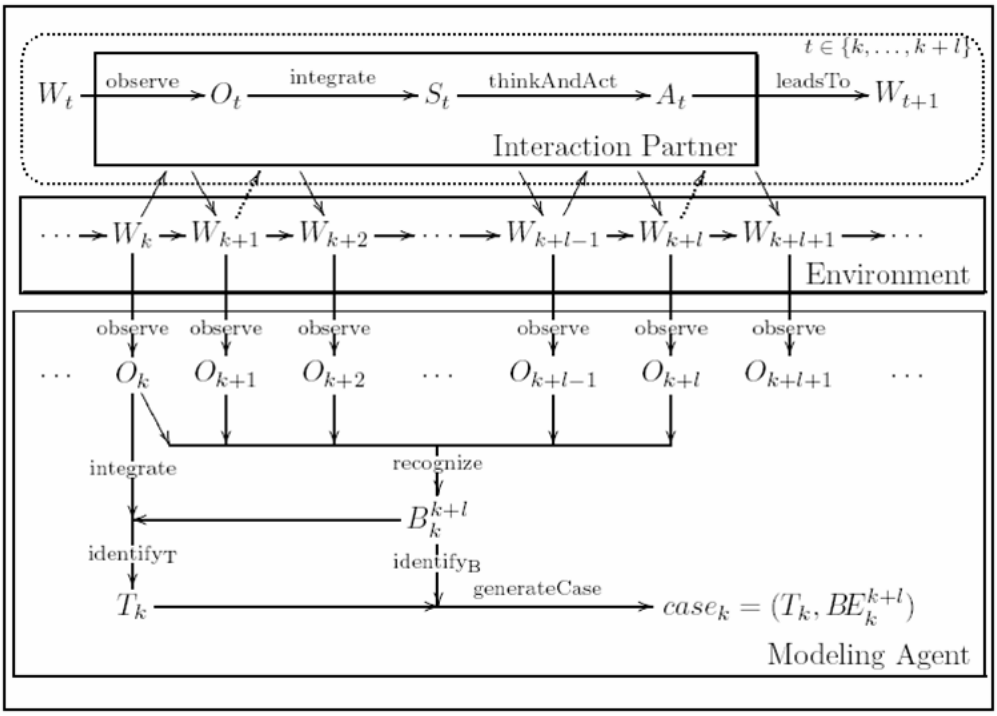


Abbildung 5.4: Ermittlung eines Auslöser-Verhaltens-Falles

Wie schon in Abschnitt 3.1.3 auf Seite 29 ausgeführt, nehmen die IP jeweils einen Auszug der Welt-Situation W_t als Beobachtung O_t wahr und reagieren mit Aktionen A_t, die zur Veränderung der Welt-Situation W_{t+1} beitragen (dargestellt im oberen Teil der Abbildung 5.4).

Damit ergibt sich eine Folge von Welt-Situationen in der Multi-Agenten-Umgebung (im mittleren Teil der Abbildung dargestellt). Die Fußball-Spieler erhalten nicht in jeder Welt-Situation eine Beobachtung. Dies ist durch die gepunkteten Pfeile beim Übergang von der Umgebung zum IP repräsentiert.

Der MA beobachtet die Multi-Agenten-Umgebung und erhält für jeden Zeittakt eine neue Beobachtung. Diese Beobachtungssequenz wertet der MA sukzessive aus, bis er ein komplettes Verhalten erkannt hat. In der Abbildung 5.4 ermittelt der MA für eine Beobachtungssequenz mit der Funktion recognize das Verhalten B_k^{k+l}, wobei k den Startzeitpunkt und $k+l$ den Endzeitpunkt des

Verhaltens darstellt. Für dieses Verhalten identifiziert der MA den Auslöser, indem er das erkannte Verhalten und die Situation zum Startzeitpunkt des Verhaltens mit der Funktion identify_T auswertet. Aus dem so ermittelten Auslöser T_k und dem erkannten Verhalten B_k^{k+l} wird mit den Funktionen identify_B und generateCase ein neuer Auslöser-Verhaltens-Fall (T_k, BE_k^{k+l}) generiert.

5.4.2 Integration eines Falles in die Fallbasis

Eine **Fallbasis** *caseBase* besteht aus einer Menge von Auslöser-Verhaltens-Fällen, $caseBase \subseteq \texttt{cases}$.

integrateCase ist die Funktion mit der ein Fall C in eine bereits existierende Fallbasis aufgenommen wird.

$$\begin{aligned} \text{integrateCase} &: 2^{\texttt{cases}} \times \texttt{cases} \rightarrow 2^{\texttt{cases}} \\ \text{integrateCase}\big(caseBase, C\big) &= caseBase \cup \{C\} \end{aligned} \tag{5.12}$$

5.5 Ähnlichkeitsmaß für die Fälle

Ein wesentlicher Aspekt beim fallbasiertem Schließen ist die Ähnlichkeit, mit deren Hilfe die Relevanz gesammelter Fälle für ein aktuelles Problem abgebildet werden soll.

Für ein gegebenes, aktuelles Problem in Form eines Auslösers sollen die Fälle in einer Fallbasis ermittelt werden, deren Verhalten dem von den IP als Reaktion auf den Auslöser tatsächlich ausgeführten Verhalten am ehesten entspricht, ihm also am ähnlichsten ist. Um das vorhergesagte Verhalten mit dem tatsächlich ausgeführten Verhalten der IP vergleichen zu können, wird ein Ähnlichkeitsmaß für die Verhalten benötigt.

In dieser Arbeit wird davon ausgegangen, daß die IP in ähnlichen Entscheidungs-Situationen, ähnliche Aktionen ausführen, so daß ähnliches beobachtbares Verhalten entsteht. Das bedeutet, daß zwei Fälle mit ähnlichen Auslösern A auch ein ähnliches Verhalten B als Lösung haben ($A_1 \sim A_2 \Rightarrow B_1 \sim B_2$). Für die Vorhersage der Verhalten der IP, müssen daher die aktuellen Auslöser mit den Auslösern der Fälle in der Fallbasis auf Ähnlichkeit verglichen werden.

Es werden also ein Ähnlichkeitsmaß zwischen Auslösern und ein Ähnlichkeitsmaß zwischen Verhalten, genauer zwischen Verhaltens-Essenzen benötigt. Dies ist in Abbildung 5.5 auf der nächsten Seite, durch die Aufteilung des Prozesses der Spezifikation der Ähnlichkeit von Fällen, dargestellt. Zwei Fälle können demnach bzgl. ihrer Auslöser sowie bzgl. ihrer Verhalten ähnlich sein.

Zuerst werden einige allgemeine Betrachtungen bzgl. der Ähnlichkeitsmaße gemacht (Abschnitt 5.5.1). Die verwendeten Ähnlichkeitsmaße sind zusammengesetzt und bestehen aus einer gewichteten Summe von lokalen Ähnlichkeiten (Abschnitt 5.5.2). Es wird ein Überblick über die verwendeten lokalen Ähnlichkeitsfunktionen gegeben (Abschnitt 5.5.3). Während bei der Ähnlichkeit von

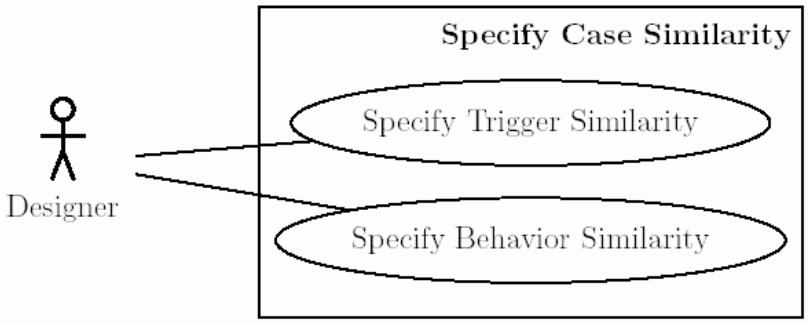


Abbildung 5.5: Struktur der Spezifikation des Ähnlichkeitsmaßes

Verhalten ein Designer die Attribute wichtet, sind die Relevanzen der Attribute bei Auslösern im allgemeinen vom jeweils modellierten Team abhängig, sie werden daher automatisch angepaßt (Abschnitt 5.5.4).

5.5.1 Allgemeine Betrachtungen

Die Auslöser und die Verhaltens-Essenzen sind jeweils durch eine Sequenz von Primärwerten und eine Sequenz von Sekundärwerten beschrieben. Die Sequenzen der Primärwerte sollen als Grundlage zur Bestimmung der Ähnlichkeit zwischen zwei Auslösern bzw. zwei Verhalten verwendet werden. Die einzelnen Werte zweier Sequenzen sollen verglichen und geeignet zu einer Gesamtähnlichkeit zusammengefasst werden.

Damit kann ein Vergleich zweier Auslöser bzw. zweier Verhalten des gleichen Musters leicht realisiert werden. Bei verschiedenen Mustern treten teilweise verschiedene Attribute auf, bei denen nicht klar ist, wie deren Werte verglichen werden sollen. Infolge der grundsätzlichen Unterschiedlichkeit der Auslöser und Verhalten verschiedener Muster wird die Ähnlichkeit bei Vergleichen zweier Auslöser bzw. zweier Verhalten verschiedener Muster mit Null definiert.

In allen anderen Fällen ist die Ähnlichkeit eine reelle Zahl zwischen Null (keine Übereinstimmung) und Eins (volle Übereinstimmung). Für Auslöser und Verhaltens-Essenzen wird je eine Funktion zur Ermittlung der Ähnlichkeit benötigt.

$\text{similarity}_\text{T}$ ist die Funktion zur Bestimmung der Ähnlichkeit zweier Auslöser desselben Auslöser-Musters.

$$\text{similarity}_\text{T} : \texttt{triggers} \times \texttt{triggers} \rightarrow \mathbb{R}_{[0,1]} \tag{5.13}$$

$\text{similarity}_\text{B}$ ist die Funktion zur Bestimmung der Ähnlichkeit zweier Verhaltens-Essenzen desselben Verhaltens-Essenz-Musters.

$$\text{similarity}_\text{B} : \texttt{behaviors}_E \times \texttt{behaviors}_E \rightarrow \mathbb{R}_{[0,1]} \tag{5.14}$$

Die in dieser Arbeit verwendeten Ähnlichkeitsfunktionen sind reflexiv, nicht symmetrisch und nicht transitiv.

5.5.2 Zusammengesetztes Ähnlichkeitsmaß

Wie im letzten Abschnitt gefordert, sollen die Ähnlichkeitsmaße zwischen Auslösern bzw. Verhalten jeweils aus den lokalen Ähnlichkeiten ihrer primären Attribute zusammengesetzt sein. Mit einer gewichteten Summe der lokalen Ähnlichkeiten wird ein einfaches zusammengesetztes Ähnlichkeitsmaß verwendet. Dabei werden musterspezifische Gewichte verwendet, d.h. die Gewichte sind für alle Verhalten bzw. Auslöser eines Musters die selben. Demnach wird für jedes Muster eine Gewichtesequenz benötigt.

$\mathtt{W}_{\mathtt{T}}^{name}$ ist die Gewichtesequenz für die Auslöser des Auslöser-Musters mit dem Namen *name*.

$$\mathtt{W}_{\mathtt{T}}^{name} \in \mathbb{R}_{+}^{n} \quad \text{mit} \quad \mathtt{W}_{\mathtt{T}}^{name} = (w_1, \ldots w_n) \quad \text{und} \quad \sum_{i=1}^{n} w_i \neq 0 \tag{5.15}$$

Die Länge der Gewichtesequenz n entspricht dabei der Länge der Primär-Attributsequenz des Auslöser-Musters $TP^{name} \in \mathtt{triggerPatterns}$, das bedeutet $TP^{name} = name\,(\langle ap_1, \ldots, ap_k\rangle, \langle as_1, \ldots, as_l\rangle)$ mit $k = n$. Die Summe aller Gewichte soll ungleich Null sein, damit bei der Bestimmung der Ähnlichkeit zweier Auslöser eine Normierung vorgenommen werden kann.

$\mathtt{W}_{\mathtt{B}}^{name}$ ist die Gewichtesequenz für die Verhaltens-Essenzen des Verhaltens-Essenz-Musters mit dem Namen *name*.

$$\mathtt{W}_{\mathtt{B}}^{name} \in \mathbb{R}_{+}^{n} \quad \text{mit} \quad \mathtt{W}_{\mathtt{B}}^{name} = (w_1, \ldots w_n) \quad \text{und} \quad \sum_{i=1}^{n} w_i \neq 0 \tag{5.16}$$

Die Länge der Gewichtesequenz n entspricht dabei der Länge der Primär-Attributsequenz des betrachteten Verhaltens-Essenz-Musters.

Die Gewichte beschreiben die Relevanz der jeweiligen Primärattribute für die Gesamtähnlichkeit zweier Auslöser bzw. zweier Verhalten. Damit können die mit Formel 5.13 und 5.14 eingeführten Ähnlichkeitsfunktionen $\text{similarity}_{\mathrm{T}}$ und $\text{similarity}_{\mathrm{B}}$ definiert werden.

$$\begin{aligned}
&\text{similarity}_{\mathrm{T}}\,(T_x, T_y)\\
&= \text{similarity}_{\mathrm{T}}\,(name\,(pSeq_x, sSeq_x), name\,(pSeq_y, sSeq_y))\\
&= \text{similarity}_{\mathrm{T}}\,(name\,(\langle x_1, \ldots, x_n\rangle, sSeq_x), name\,(\langle y_1, \ldots, y_n\rangle, sSeq_y))\\
&= \frac{1}{\sum_{i=1}^{n} w_i} * \sum_{i=1}^{n} w_i * \text{sim}_{\mathtt{T},i}^{name}(x_i, y_i) \qquad \text{mit} \quad \mathtt{W}_{\mathtt{T}}^{name} = (w_1, \ldots w_n)\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad T_x, T_y \in \mathtt{triggers}
\end{aligned} \tag{5.17}$$

Mit der Division durch die Summe der Gewichte eines Auslöser-Musters wird das Ergebnis normiert, so daß die Ähnlichkeit immer einen Wert zwischen Null und Eins hat.

$\mathrm{sim}_{\mathtt{T},i}^{name}$ ist eine lokale Ähnlichkeitsfunktion, die für zwei Werte desselben Attributes $\mathbb{A}_i$ definiert ist.

$$\mathrm{sim}_{\mathtt{T},i}^{name} : \mathbb{A}_i \times \mathbb{A}_i \to \mathbb{R}_{[0,1]} \tag{5.18}$$

Die Mengen $\mathbb{A}_i$ bezeichnen dabei eine der Attributmengen aus Tabelle 5.1. Die lokalen Ähnlichkeitsfunktionen werden im nächsten Abschnitt genauer betrachtet.

Die Ähnlichkeitsfunktion $\mathrm{similarity}_{\mathrm{B}}$ wird analog der Funktion $\mathrm{similarity}_{\mathrm{T}}$ definiert. Statt der Gewichtesequenzen $\mathtt{W}_{\mathtt{T}}^{name}$ werden die Gewichtesequenzen $\mathtt{W}_{\mathtt{B}}^{name}$ verwendet und statt der lokalen Ähnlichkeitsfunktionen $\mathrm{sim}_{\mathtt{T},i}^{name}$ werden die Funktionen $\mathrm{sim}_{\mathtt{B},i}^{name}$ verwendet. Die Funktionen $\mathrm{sim}_{\mathtt{B},i}^{name}$ stellen dabei wieder lokale Ähnlichkeitsfunktionen dar.

Die Gewichtesequenzen für die Verhaltens-Essenzen $\mathtt{W}_{\mathtt{B}}^{name}$ werden durch den Designer vorgegeben. Für die Auslöser werden die Gewichtesequenzen dagegen automatisch angepaßt. Dies wird in Abschnitt 5.5.4 behandelt.

5.5.3 Lokale Ähnlichkeiten

Die Ähnlichkeit zwischen den Werten eines Attributes wird durch den Designer festgelegt. Damit muß für jedes primäre Attribut eine Ähnlichkeitsfunktion definiert werden. Anhand ihrer Arten von Wertebereichen können die primären Attribute in vier verschiedene Klassen unterteilt werden. Diese sind in Tabelle 5.3 dargestellt. In der linken Spalte ist die Attributklasse benannt, die mittlere Spalte beschreibt die Eigenschaften der Mengen der zur jeweiligen Klasse gehörenden Attribute und in der dritten Spalte sind die zur jeweiligen Attributklasse gehörenden Attributmengen aufgezählt. Die Attributmengen wurden in den Tabellen 5.1 auf Seite 109 und 5.2 auf Seite 113 eingeführt.

Attributklasse	Eigenschaft	Attributmengen
Vektoren	Teilmenge von $\mathbb{V}$	$\mathbb{KP}$, $\mathbb{RP}$, $\mathbb{PM}$
reelle Zahlen	Teilmenge von $\mathbb{R}$	$\mathbb{OA}$, $\mathbb{OD}$, $\mathbb{SL}$, $\mathbb{SA}$
natürliche Zahlen	Teilmenge von $\mathbb{N}$	$\mathbb{DU}$
Aufzählungen	endliche Menge	$\mathbb{VD}$, $\mathbb{SU}$

Tabelle 5.3: Die vier Attributtypen

Für die Teilmengen der Vektoren, der reellen und der natürlichen Zahlen wird die Ähnlichkeit über den Abstand ihrer Werte definiert, je größer der Abstand desto geringer die Ähnlichkeit und umgekehrt. Bei reellen und natürlichen Zahlen wird als „Abstand“ die Differenz[6] und bei Vektoren der euklidische Abstand verwendet. Ab einem für das jeweilige Attribut ausgezeichneten Abstand wird die Ähnlichkeit auf Null gesetzt, bei sehr geringen Abständen wird

[6]Insbesondere wird zwischen positiven und negativen „Abständen“ einiger Attribute unterschieden, so daß ein negativer und ein positiver „Abstand“ mit gleichem Betrag zu unterschiedlichen Ähnlichkeiten führen kann. Damit ist das verwendete Ähnlichkeitsmaß nicht symmetrisch.

die Ähnlichkeit dagegen auf Eins gesetzt. Bei Aufzählungsmengen wird die Ähnlichkeit für alle Wertepaare speziell definiert. Beispielhaft wird die Ähnlichkeitsfunktion für die 'Positionen zweier Kicker' angegeben.

$\mathrm{sim}^{\mathrm{pass}}_{\mathtt{T},1}$ ist die Funktion, die die Ähnlichkeit zweier Kickerpositionen des Auslöser-Musters Paß bestimmt.

$$\mathrm{sim}^{\mathrm{pass}}_{\mathtt{T},1} : \mathbb{KP} \times \mathbb{KP} \rightarrow \mathbb{R}_{[0,1]}$$

$$\mathrm{sim}^{\mathrm{pass}}_{\mathtt{T},1}(v_x, v_y) = \begin{cases} 1 & \text{falls } (v_y - v_x).length < 2 \\ 0 & \text{falls } (v_y - v_x).length > 15 \\ 1 - \frac{(v_y - v_x).length - 2}{15 - 2} & \text{sonst} \end{cases} \tag{5.19}$$

Für die Kickerpositionen der anderen Auslöser-Muster ist die Ähnlichkeitsfunktion identisch definiert.

5.5.4 Automatische Bestimmung von Gewichten

Bei der Spezifikation der Auslöser durch einen Designer können nur die wahrscheinlich relevanten Attribute für das Zustandekommen der entsprechenden Verhalten ermittelt werden. Ob diese Attribute tatsächlich relevant für das Zustandekommen von Verhalten sind, kann nicht gesagt werden. So ist es durchaus möglich, daß für einige Teams nicht alle der ausgewählten primären Attribute relevant sind. Desweiteren kann es sein, daß die Attribute unterschiedlich starken Einfluß auf das Zustandekommen von Verhalten haben. Es ist vorstellbar, daß ein Team mehr Wert auf den Winkelabstand zum nächsten Gegner legt, ein anderes Team den Vektor zum Empfänger stärker wichtet. Die Relevanz der einzelnen primären Attribute hängt demnach vom modellierten Team ab.

Gegenstand dieses Abschnittes ist es, die Relevanzen der primären Attribute der Auslöser-Muster automatisch zu ermitteln. Dabei beschränken wir uns darauf, die Gewichtesequenzen der Auslöser $\mathtt{W}^{name}_{\mathtt{T}}$ an das jeweils modellierte Team anzupassen. Für jedes Auslöser-Muster soll aus einer Mengen von Gewichtesequenzen die Sequenz ausgewählt werden, die die besten Vorhersageresultate liefert.

Bei der automatischen Bestimmung der Gewichtesequenzen wird das Verfahren des *Cross Validation* verwendet, welches im folgenden vorgestellt wird. Danach wird es für die Bestimmung der Gewichte für Auslöser angewendet. Schließlich werden Bemerkungen zur Auswahl von geeigneten Primärdaten für die Anwendung des *Cross Validation* gemacht.

5.5.4.1 Das Verfahren des *Cross Validation*

Beim *Cross Validation* [31, 42] geht es darum, aus einer Menge von Modellen $\mathbb{M}$ auf Basis einer endlichen Datenmenge $\mathbb{D} \subset \mathbb{X} \times \mathbb{Y}$, das Modell $M^* \in \mathbb{M}$ zu bestimmen, welches eine funktionale Abhängigkeit zwischen den Eingabewerten $X \in \mathbb{X}$ und den Ausgabewerten $Y \in \mathbb{Y}$ am besten beschreibt. Das Besondere beim *Cross Validation* ist, daß die Datenmenge $\mathbb{D}$ in eine Basisdaten- und

Testdaten-Menge unterteilt wird, wobei jedes Modell in Kombination mit der Basisdaten-Menge mit jedem Testdatum bewertet wird. Auf Basis dieser Bewertungen wird das am besten geeignete Modell M^* bestimmt. Die Aufteilung der Datenmenge $\mathbb{D}$ in die Basisdaten- und Testdaten-Menge kann dabei zwischen den einzelnen Bewertungen varieren.

Cross Validation ist ein häufig angewandtes Verfahren zur Modellauswahl. Es ist ein sehr allgemeines Verfahren, unterliegt nach [42] aber auch einigen Einschränkungen:

- einige Daten von $\mathbb{D}$ müssen als Testmenge reserviert werden,
- bei zu großer Menge von Modellen kann das ausgewählte Modell M^* übergenau sein, das bedeutet es spiegelt exakt die verwendete Datenmenge wieder, ist aber schlecht zur Vorhersage für weitere Daten geeignet (im englischen spricht man von *Overfitting*),
- die Bestimmung einer ausreichend repräsentativen Menge von Daten, für eine gegebene Menge von Modellen ist extrem schwierig,
- die Bestimmung des besten Modells M^* aus einer großen Menge von Modellen kann sehr zeitaufwendig sein.

Bei genauerer Betrachtung des *Cross Validation* Verfahren lassen sich folgende Bestandteile ableiten:

- ein Definitionsbereich $\mathbb{X}$ und ein Wertebereich $\mathbb{Y}$,
- eine Menge von Modellen $\mathbb{M} = \{M_1, \dots M_n\}$,
- eine endliche Menge von Basisdaten $\mathbb{BD} \subset \mathbb{X} \times \mathbb{Y}$,
- eine endliche Menge von Testdaten $\mathbb{TD} \subset \mathbb{X} \times \mathbb{Y}$,
- ein Algorithmus $\mathrm{p} : \mathbb{M} \times 2^{(\mathbb{X}\times\mathbb{Y})} \times \mathbb{X} \to \mathbb{Y}$, der basierend auf einem Modell und einer Menge von Basisdaten, die funktionale Abhängigkeit zwischen den Ein- und Ausgabewerten beschreibt,
- eine Funktion $\mathrm{e} : \mathbb{M} \times \mathbb{TD} \to \mathbb{R}_+$, die den Fehler eines Modells $M_i \in \mathbb{M}$ bei Verwendung des Datums $(X_j, Y_j) \in \mathbb{TD}$ berechnet

$$\mathrm{e}(M_i, (X_j, Y_j)) = \|Y_j - \mathrm{p}(M_i, \mathbb{BD} \setminus \{(X_j, Y_j)\}, X_j)\| \tag{5.20}$$

- und eine Funktion $\mathrm{e}^* : \mathbb{M} \to \mathbb{R}_+$, die den durchschnittlichen Fehler eines Modells $M_i \in \mathbb{M}$ für eine Menge von Testdaten $\mathbb{TD}$ bestimmt.

$$\mathrm{e}^*(M_i) = \frac{1}{|\mathbb{TD}|} \sum_{(X_j,Y_j)\in\mathbb{TD}} \mathrm{e}(M_i, (X_j, Y_j)). \tag{5.21}$$

Da die Mengen $\mathbb{BD}$ und $\mathbb{TD}$ nicht notwendigerweise disjunkt sind, wird bei der Berechnung des Fehlers durch die Funktion e das Datum (X_j, Y_j) aus der Menge der Basisdaten entfernt (falls es sich in dieser befindet)[7]. Die Differenz

[7] Ein häufig verwendeteter Spezialfall des *Cross Validation* tritt ein, wenn die Menge der Basisdaten der Menge der Testdaten entspricht. Dieser Speziallfall wird mit *leave one out cross validation* bezeichnet.

des Ergebnisses des Algorithmus und des tatsächlichen Werts Y_j wird bestimmt und daraus ein Fehlerwert ermittelt.

Beim *Cross Validation* ist das Modell M^* gesucht, welches den durchschnittlichen Fehler $\mathrm{e}^*(M_i)$ minimiert.

$$M^* = \underset{M_i \in \mathbb{M}}{\operatorname{argmin}}\ \mathrm{e}^*(M_i) \tag{5.22}$$

Der Algorithmus p ist meist sehr zeitaufwendig, so umfasst er z.B. in [31] das automatische Lernen eines Entscheidungsbaums mittels des Algorithmus *C4.5* [48]. Für jedes Modell M_i muß dort der Entscheidungsbaum neu gelernt werden. Damit besteht ein hoher Bedarf an einer Optimierung der Suche nach dem besten Modell M^*, bzw. einer Begrenzung der zu untersuchenden Modelle.

Für spezielle Modell-Mengen $\mathbb{M}$ existieren optimierte Verfahren zur Suche nach dem besten Modell M^*, die es nicht nötig machen, alle Modelle zu untersuchen. In dieser Arbeit wird versucht die zu untersuchenden Modelle im Vorfeld zu reduzieren.

5.5.4.2 Anwendung des *Cross Validation*

Zur Anwendung des *Cross Validation* müssen die im letzten Abschnitt vorgestellten Bestandteile des Verfahrens für das automatische Ermitteln von Gewichtesequenzen für Auslöser identifiziert werden. Dazu werden zuerst noch nicht vorhandene Mengen und Funktionen eingeführt.

Als Modelle werden die Gewichtesequenzen der Auslöser-Muster verwendet. Da die Gewichtesequenzen aller Auslöser-Muster angepaßt werden sollen, wird eine alle Auslöser umfassende Gewichtesequenz $\mathtt{W_T}$ benötigt. Diese erhalten wir durch Verkettung $\mathtt{W_T} = (\mathtt{W_T^{pass}}, \mathtt{W_T^{scoreGoal}}, \mathtt{W_T^{dribble}})$. Beim *Cross Validation* werden verschiedene solcher Gewichtesequenzen evaluiert. Die zu untersuchenden Gewichtesequenzen werden in einer Menge $\mathtt{weights_T}$ zusammengefasst.

$\mathtt{weights_T}$ ist die Menge der Gewichtesequenzen $\mathtt{W_T}$ aller Auslöser-Muster, aus denen beim *Cross Validation* die optimale Gewichtesequenz ermittelt werden soll.

$$\mathtt{weights_T} \subseteq \mathbb{R}_+^n \tag{5.23}$$

Dabei ist n die Summe der Längen der Primär-Attributsequenzen aller Auslöser-Muster. Bei Betrachtung der Auslöser-Muster Paß, Torschuß und Dribbeln hat n damit den Wert von $6 + 7 + 7 = 20$.

Damit genauere Angaben über den Vorhersage-Algorithmus p gemacht werden können, wird die Funktion zur Vorhersage von Verhalten bereits in diesem Abschnitt eingeführt.

predict ist die Verhaltens-Vorhersage Funktion, die auf Basis einer Gewichtesequenz für Auslöser $W \in \mathtt{weights_T}$ und einer Fallbasis $caseBase \subset \mathtt{cases}$ für eine Situation $S \in \mathtt{situations}$ das Verhalten $BE \in \mathtt{behaviors}_E$ der Interaktionspartner vorhersagt.

$$\text{predict} : \mathtt{weights_T} \times 2^{\mathtt{cases}} \times \mathtt{situations} \rightarrow \mathtt{behaviors}_E \tag{5.24}$$

Diese Funktion wird genauer in Abschnitt 6.1.6 auf Seite 135 behandelt.

Nun werden die Bestandteile des *Cross Validation* Verfahrens zur Bestimmung der Gewichtesequenzen von Auslöser-Mustern herausgearbeitet.

- Als Definitionsbereich wird die Menge der Situationen, $\mathbb{X} = \texttt{situations}$ und als Wertebereich die Menge der Verhaltens-Essenzen verwendet, $\mathbb{Y} = \texttt{behaviors}_E$.
- Als Menge von Modellen wird die Menge der Gewichtesequenzen verwendet, $\mathbb{M} = \texttt{weights}_\texttt{T}$.
- Als Basisdaten stehen die erkannten Verhaltens-Essenzen eines Teams und deren jeweilige Entscheidungs-Situationen von mehreren Spielen zur Verfügung, $\mathbb{BD} \subset \texttt{situations} \times \texttt{behaviors}_E$.
- Als Testdaten werden alle erkannten Verhaltens-Essenzen eines Teams und deren jeweilige Entscheidungs-Situationen eines Spieles verwendet, $\mathbb{TD} \subset \texttt{situations} \times \texttt{behaviors}_E$.
- Der Algorithmus p wird über die Verhaltens-Vorhersage Funktion predict definiert.

$$\mathrm{p}(W, \mathbb{BD} \setminus \{(S_j, BE_j)\}, S_j) = \mathrm{predict}(W, caseBase, S_j) \quad \text{mit } (S_j, BE_j) \in \mathbb{TD} \tag{5.25}$$

  Dabei wird die Menge $caseBase \in \texttt{cases}$ aus der Menge $\mathbb{BD} \setminus \{(S_j, BE_j)\}$ mit den Funktionen $\mathrm{identify_T}$, generateCase und integrateCase erzeugt (vgl. Formeln 5.5, 5.11 und 5.12).

- Die Funktion e ergibt sich aus der Ähnlichkeit zwischen dem aus der Situation S vorhergesagten Verhalten und dem tatsächlich von den IP ausgeführten Verhalten BE.

$$\mathrm{e}(W_i, (S_j, BE_j)) \;=\; 1 - \mathrm{similarity_B}\left(BE_j, \mathrm{predict}(W_i, caseBase, S_j)\right) \tag{5.26}$$

- Bei der Funktion e^* ändern sich nur die Bezeichnungen.

Damit ist das Verfahren zur Bestimmung der optimalen Gewichtesequenz $W^* \in \texttt{weights}_\texttt{T}$ für Auslöser-Muster vollständig. Aus den Formeln 5.21 und 5.22 folgt:

$$W^* = \underset{W_i \in \texttt{weights}_\texttt{T}}{\operatorname{argmin}} \frac{1}{|\mathbb{TD}|} \sum_{(S_j, BE_j) \in \mathbb{TD}} \mathrm{e}(W_i, (S_j, BE_j)). \tag{5.27}$$

Nach dem Einfügen von Formel 5.26 und einer Umformung ergibt sich:

$$W^* = \underset{W_i \in \texttt{weights}_\texttt{T}}{\operatorname{argmax}} \frac{1}{|\mathbb{TD}|} \sum_{(S_j, BE_j) \in \mathbb{TD}} \mathrm{similarity_B}\left(BE_j, \mathrm{predict}(W_i, caseBase, S_j)\right). \tag{5.28}$$

Das bedeutet, daß die Gewichtesequenz bestimmt wird, die die durchschnittliche Ähnlichkeit zwischen vorhergesagten Verhalten und erkannten Verhalten für eine Testmenge $\mathbb{TD}$ maximiert. Die Gewichtesequenz wird in Abhängigkeit von den Basisdaten und Testdaten bestimmt. Damit ist die Sequenz vom modellierten Team abhängig.

Die Komplexität der Bestimmung der optimalen Gewichtesequenz W^* ist jeweils linear von der Kardinalität der Menge der Testdaten $\mathbb{TD}$, von der Kardinalität der Menge der Gewichtesequenzen $\texttt{weights}_\texttt{T}$ und von der Komplexität der Verhaltens-Vorhersage Funktion predict abhängig. Da die Berechnung mit der Funktion predict bereits zeitaufwendig ist[8], kommt der Begrenzung der Testdaten und besonders der Begrenzung der Menge der Gewichtesequenzen eine besondere Bedeutung zu. Die Begrenzung der Menge der Gewichtesequenzen wird u.a. im nächsten Abschnitt betrachtet.

5.5.4.3 Bemerkungen zur Wahl der Basis-, Test- und Gewichtsmengen

Durch die Abhängigkeit der zu ermitteltenden Gewichtesequenz von den Basis- und Testdaten ergibt sich die Forderung, daß diese repräsentativ für zukünftige Entscheidungs-Situationen sein müssen. Andernfalls ist es möglich, daß eine Gewichtesequenz bestimmt wird, die zwar für die Menge der Testdaten optimal ist, aber bei anderen Entscheidungs-Situationen schlechte Vorhersage-Ergebnisse erzielt. Es ist anzunehmen, daß bei der Verwendung von Basis- und Testdaten, die aus realen Spielen stammen, eine repräsentative Auswahl getroffen wird.

Bei der Auswahl der Menge der mit *Cross Validation* zu untersuchenden Gewichtesequenzen $\texttt{weights}_\texttt{T}$ ist eine geeignete Beschränkung nötig. So besteht bei einer zu großen Menge $\texttt{weights}_\texttt{T}$ eine erhöhte Gefahr von *Overfitting*. Desweiteren kann die Bestimmung der besten Gewichtesequenz aus einer großen Menge von Gewichtesequenzen sehr zeitaufwendig sein.

Für die Beschränkung der Menge $\texttt{weights}_\texttt{T}$ treffen wir die Annahme, daß gleiche Attribute $\mathbb{A}_i$ jeweils die gleiche Relevanz haben, d.h. daß z.B. die Kickerpositionen in allen betrachteten Auslöser-Mustern das gleiche Gewicht erhalten. Dies reduziert den potentiellen Zustandsraum der Menge $\texttt{weights}_\texttt{T}$ bei Verwendung der betrachteten Auslöser-Muster (deren Primär-Attributesequenzen in der Summe eine Länge von 20 haben) von $\mathbb{R}_+^{20}$ auf $\mathbb{R}_+^5$, da nur 5 verschiedene primäre Attribute existieren (vgl. oberer Teil von Tabelle 5.1 auf Seite 109).

Es genügt, sich bei den Gewichten auf natürliche Zahlen zwischen 0 und n zu begrenzen. Wenn die Gewichte jedes der Attribute zwischen 0 und n variiert werden, ergeben sich $(n+1)^5$ Gewichtesequenzen. Nur für sehr kleine n kann eine solche Menge von Gewichtesequenzen noch mit dem vorgestellten Verfahren und vertretbaren Aufwand untersucht werden.

[8] Bei knapp über 1000 Fällen benötigt die Funktion predict bei Verwendung eines AMD-Prozessors mit 400 MHz eine Zeit von circa 60 Millisekunden.

Statt einer Variation der Gewichte zwischen 0 und n wird eine Verteilung von n Gewichtseinheiten auf k primäre Attribute betrachtet, d.h. n „Gewichtspunkte“ werden vollständig auf die k Attribute verteilt, $\texttt{weights}_\texttt{T} = \{(w_1, \ldots, w_k) : w_i \in \mathbb{N} \wedge \sum_{i=1}^{k} w_i = n\}$. Damit hat die Menge der Gewichtesequenzen $\binom{n+(k-1)}{k-1}$ Elemente. Für 10 Gewichte und 5 primäre Attribute ergeben sich dann $\binom{10+(5-1)}{5-1} = 1001$ Gewichtesequenzen.

Bei Verwendung eines AMD-Prozessors mit 400 MHz, 1000 Fällen und einer Testmenge mit einer Kardinalität von 100 benötigt der Rechner für die Bestimmung der optimalen Gewichtesequenz aus einer Menge von 1001 Gewichtesequenzen circa 6000 Sekunden, also knapp 2 Stunden. Das bedeutet, daß eine Bestimmung der Gewichtesequenzen nur ‘offline’ nach der Analyse mehrerer Spiele sinnvoll ist.

5.6 Objektorientierte Realisierung

In diesem Abschnitt wird eine objektorientierte Realisierung der Verhaltens-Modellierung, genauer der Generierung von Verhaltens-Modellen, präsentiert. Dazu werden die in diesem Kapitel vorgestellten Mengen und Funktionen auf Klassen und Methoden abgebildet.

Abbildung 5.6 auf der nächsten Seite zeigt eine Übersicht der zur Verhaltens-Modellierung verwendeten Klassen. Die wichtigsten Bestandteile der Objekte dieser Klassen und die Beziehungen zwischen den Klassen sind dargestellt. Ein **Teammodel** enthält eine Fallbasis *caseBase*, die aus einer Menge von Fällen besteht. Ein Fall (repräsentiert durch ein Objekt der Klasse **Case**) besteht aus einem Auslöser *trigger* und einer Verhaltens-Essenz *behaviorE*. Die Gemeinsamkeiten aller Auslöser und Verhaltens-Essenzen sind in der Klasse **Concept** zusammengefasst. So besteht jeder Auslöser und jede Verhaltens-Essenz aus einer Sequenz von Primärwerten *primaryValues* und aus einer Menge von Sekundärwerten *secondaryValues*, die jeweils aus einer Menge von Objekten der Klasse **Attribute** bestehen. Die Klassen **Trigger** und **BehaviorEssence** sind abstrakt. Von ihnen werden alle konkreten Auslöser bzw. Verhaltens-Essenzen abgeleitet.

Es ergeben sich vier Gruppen von Klassen die im folgenden genauer vorgestellt werden. Es wird die objektorientierte Realisierung von Attributen (Abschnitt 5.6.1), von Auslösern und Verhaltens-Essenzen (Abschnitt 5.6.2), von Fällen (Abschnitt 5.6.3) und von Teammodellen (Abschnitt 5.6.4) behandelt.

In diesen Abschnitten wird die Generierung und die Speicherung von Objekten sowie die Bestimmung der Ähnlichkeit und Identität zwischen Objekten betrachtet. Bei der Speicherung werden die Fälle in Zeichenketten umgewandelt und in eine externe Datenbasis (Datei) geschrieben, die bei einem erneuten Start des Coach-Programmes eingelesen werden kann. Damit beim Verfahren des *Cross Validation* mögliche identische Fälle aus der Fallbasis entfernt werden können, ist neben der Ähnlichkeit auch die Identität zweier Fälle zu behandeln.

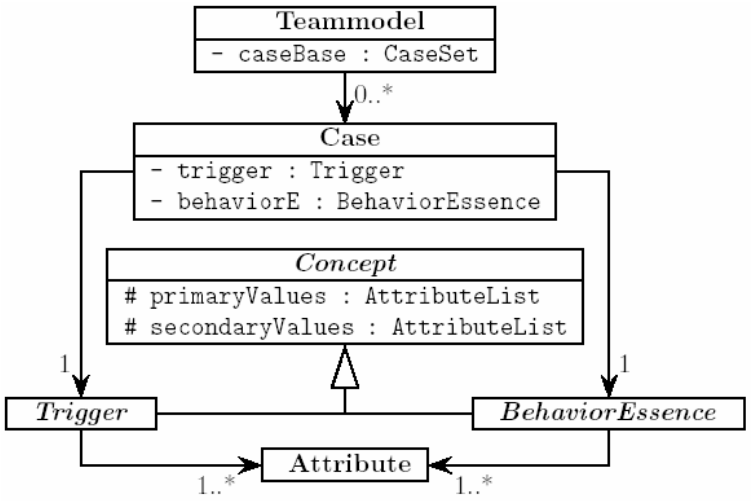


Abbildung 5.6: Klassenstruktur eines Teammodels

5.6.1 Attribute von Auslöser und Verhalten

Für jedes der benötigten Attribute aus den Tabellen 5.1 auf Seite 109 und 5.2 auf Seite 113 ist es möglich, eine eigene Klasse einzuführen. Aus Gründen der Übersichtlichkeit ist es aber eher ratsam, alle Attribute in einer Klasse **Attribute** zusammenzufassen. Damit besteht ein Attribut aus einem Typbezeichner *type* und einem Wert *value*, die bei der Generierung festgelegt werden. Gültige Typbezeichner sind in der Klasse **AttributeType** zusammengefasst.

In Abbildung 5.7 sind die Klassen **Attribute** und **AttributeType** dargestellt. Mit der Methode *similarityTo* wird die Ähnlichkeit zweier Attribute desselben Typs bestimmt. Damit werden die lokalen Ähnlichkeitsfunktionen $\mathrm{sim}_{\mathtt{T},i}^{name}$ bzw. $\mathrm{sim}_{\mathtt{B},i}^{name}$ realisiert (vgl. Formeln 5.18 und 5.19).

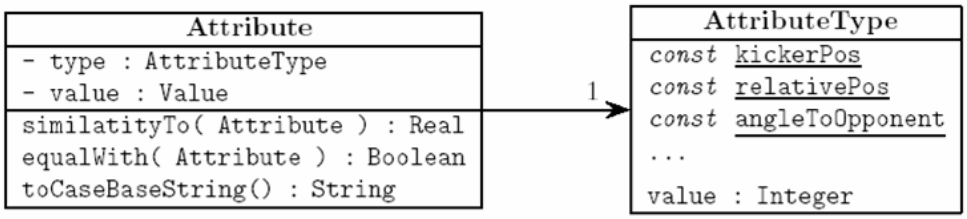


Abbildung 5.7: Klassendiagramm für die Attribute

Die Methode *equalWith* dient zum Test auf Identität zweier Attribute. Zwei Attribute sind identisch, wenn sie den selben Typ haben und ihre Werte gar nicht oder nur sehr gering voneinander abweichen[9]. Mit der Methode *toCaseBaseString* wird das Attribut in eine Zeichenkette umgewandelt.

[9]Geringe Abweichungen werden zugelassen, damit trotz Rundungsfehler beim Speichern in der Datenbasis, identische Auslöser erkannt werden können.

5.6.2 Auslöser und Verhalten

Auslöser und Verhaltens-Essenzen weisen einige Gemeinsamkeiten auf, so bestehen alle Auslöser und Verhaltens-Essenzen jeweils aus einer Sequenz von Primärwerten und aus einer Sequenz von Sekundärwerten. Die Ähnlichkeitsberechung wie auch andere Methoden erfolgen nach dem selben Schema. Die gemeinsamen Daten und die gemeinsame Funktionalität von Auslösern und Verhaltens-Essenzen werden daher in einer gemeinsamen, abstrakten Superklasse **Concept** zusammengefasst.

Von der Klasse **Concept** sind die abstrakten Klassen **Trigger** und **BehaviorEssence** abgeleitet. Die Klasse **Trigger** ist eine Generalisierung aller Auslöser und damit die Superklasse der Klassen **PassTrigger**, **ScoreGoalTrigger** und **DribbleTrigger**. Die Klasse **BehaviorEssence** ist eine Generalisierung aller Verhaltens-Essenzen, also die Superklasse der Klassen **PassBehaviorEssence**, **ScoreGoalBehaviorEssence** und **DribbleBehaviorEssence**. Damit ist jedes Auslöser- bzw. Verhaltens-Essenz-Muster durch eine Klasse repräsentiert, deren Objekte Auslöser bzw. Verhaltens-Essenzen darstellen. Die Beziehungen zwischen den Klassen sind in Abbildung 5.8 wiedergegeben.

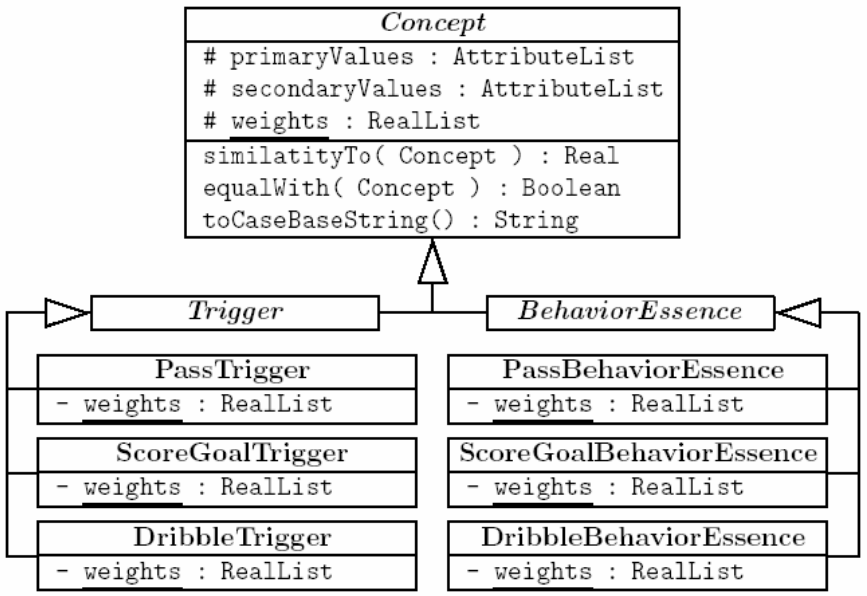


Abbildung 5.8: Klassendiagramm für die Auslöser und die Verhalten

Ein Auslöser wird auf Basis einer Entscheidungs-Situation und eines Zieles (ein Spieler, das Tor oder eine virtuelle Richtung) generiert. Das Ziel kann dabei das Ziel eines erkannten Verhaltens oder ein potentielles Ziel einer Situation sein. Die Generierung einer Verhaltens-Essenz erfolgt auf Basis eines erkannten Verhaltens. Bei der Generierung eines Auslösers bzw. einer Verhaltens-Essenz werden die Wertesequenzen der primären und der sekundären Attribute erzeugt. Die Wertesequenzen sind durch die Listen *primaryValues* und *secondaryValues* gegeben.

Mit der Methode *similarityTo* kann die Ähnlichkeit zwischen zwei Auslösern bzw. Verhaltens-Essenzen bestimmt werden. Zur Bestimmung der Ähnlichkeit werden die primären Wertesequenzen *primaryValues* der beiden Objekte paarweise verglichen. Die jeweiligen Resultate der einzelnen Wertepaare werden gewichtet und aufsummiert. Die dafür verwendete Gewichtesequenz ist für jedes Auslöser- bzw. Verhaltens-Essenz-Muster spezifisch und wird deshalb mit der Klassenvariablen *weights* realisiert. Somit sind die Ähnlichkeitsfunktionen $\text{similarity}_\text{T}$ bzw. $\text{similarity}_\text{B}$ definiert (vgl. Formeln 5.13, 5.14 und 5.17).

Beim Test auf Identität zweier Auslöser bzw. Verhaltens-Essenzen mit der Methode *equalWith* werden deren primäre und sekundäre Wertesequenzen verglichen. Wenn diese paarweise identisch sind, dann gelten auch die Auslöser bzw. Verhaltens-Essenzen als identisch.

Bei der Speicherung eines Objektes der Klasse **Concept** mit der Methode *toCaseBaseString* werden die Sequenzen *primaryValues* und *secondaryValues* in eine Zeichenkette umgewandelt. Ein das Auslöser- bzw. Verhaltens-Muster eindeutig bezeichnender Identifikator wird als Bestandteil dieser Zeichenkette aufgenommen.

5.6.3 Fälle

Mit einem Auslöser *trigger* und der dazugehörigen Verhaltens-Essenz *behaviorE* kann ein Fall entsprechend der Funktion generateCase (vgl. Formel 5.11) erzeugt werden. Die wesentlichen Eigenschaften eines Falles sind in Abbildung 5.9 dargestellt.

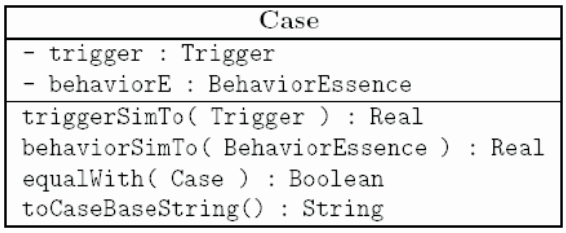


Abbildung 5.9: Klassendiagramm für die Fälle

Ein Fall muß sowohl mit einem Auslöser, als auch mit einer Verhaltens-Essenz auf Ähnlichkeit verglichen werden können. Die Bestimmung der Ähnlichkeit eines Falles mit einem Auslöser wird mit der Methode *triggerSimTo* durchgeführt, für die Ermittlung der Ähnlichkeit zwischen Fall und Verhaltens-Essenz ist die Methode *behaviorSimTo* zuständig. In beiden Fällen wird die Methode *similarityTo* des Auslösers bzw. der Verhaltens-Essenz genutzt.

Zwei Fälle sind bzgl. der Methode *equalWith* identisch, wenn deren Auslöser und Verhaltens-Essenzen jeweils identisch sind. Bei der Speicherung mit der Methode *toCaseBaseString* werden Auslöser und Verhaltens-Essenz jeweils in eine Zeichenkette transformiert und miteinander verbunden.

5.6.4 Teammodel

Die Beschreibung des Verhaltens eines Teams erfolgt durch die Klasse **Teammodel**, dessen wichtigste Bestandteile bzgl. der Generierung von Verhaltens-Modellen in Abbildung 5.10 dargestellt sind. Die Funktionalität dieser Klasse bzgl. der Verhaltens-Vorhersage und der Aktualisierung von Verhaltens-Modellen wird erst später, in Abschnitt 6.3 auf Seite 138, betrachtet.

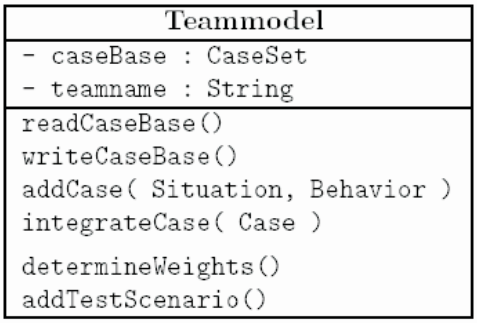


Abbildung 5.10: Klassendiagramm eines Teammodels

Jedes Teammodel ist einem Team zugeordnet, dessen Name mit der Zeichenkette *teamname* festgehalten wird. Die Fallbasis *caseBase* eines Teammodels ist anfänglich leer. Bei der Generierung des Teammodels werden die Daten eines in einer Datei gespeicherten Teammodels mit der Methode *readCaseBase* eingelesen. Zu diesen Daten gehören die Auslöser-Verhaltens-Fälle und die Gewichtesequenzen der einzelnen Auslöser- und Verhaltens-Muster.

Mit jedem erkannten Verhalten kann das Teammodel mit der Methode *addCase* um einen Fall erweitert werden. Dafür müssen die Situation (Objekt der Klasse **Situation**) und das erkannte Verhalten (Objekt der Klasse **Behavior**) übergeben werden. Aus diesen beiden Objekten wird ein Auslöser und eine Verhaltens-Essenz erzeugt, mit denen ein neuer Fall generiert wird. Dieser Fall wird mit der Methode *integrateCase* in die Fallbasis *caseBase* aufgenommen.

Damit das generierte Teammodel auch für spätere Untersuchungen zur Verfügung steht, wird es in einer Datei gespeichert. Dazu werden sowohl die Fälle, die Gewichtesequenzen, der Teamname des modellierten Teams als auch weitere Daten mit der Methode *writeCaseBase* in eine Datei geschrieben.

Die Bestimmung der Gewichtesequenzen der Auslöser-Muster kann mit der Methode *determineWeights* erfolgen. Dazu wird jedes erkannte Situation-Verhaltens-Paar mit der Methode *addTestScenario* festgehalten. Nach Analyse des Spieles werden diese Testdaten verwendet um mit der Methode *determineWeights* die Gewichtesequenzen der Auslöser-Muster anzupassen.

Das Teammodel kann an mehreren Stellen konfiguriert werden. Es kann festgelegt werden, für welche der Teams ein Verhaltens-Modell generiert werden soll. Das Erzeugen, Einlesen und Speichern von Fällen sowie die Bestimmung der Gewichtesequenzen kann ein- bzw. abgeschaltet werden. Desweiteren kann die Anzahl der zu verwendenden „Gewichtspunkte“ für die Bestimmung von Gewichtesequenzen (vgl. Abschnitt 5.5.4.3 auf Seite 124) konfiguriert werden.

Kapitel 6

Anwendung von Verhaltens-Modellen

Im Rahmen dieser Arbeit werden fallbasierte Verhaltens-Modelle für die Vorhersage von Verhalten und bei der Aktualisierung von Verhaltens-Modellen angewendet. Für Entscheidungs-Situationen wird das zukünftigen Verhalten der Interaktionspartner vorhergesagt. Dazu wird in mehreren Schritten ein für die Vorhersage geeigneter Auslöser-Verhaltens-Fall bestimmt, dessen Lösung dann an die Entscheidungs-Situation angepaßt wird. Bei einer fehlerhaften Vorhersage wird das Verhaltens-Modell durch die Aufnahme eines neuen Auslöser-Verhaltens-Falles aktualisiert.

Zuerst werden die Teilprozesse der Verhaltens-Vorhersage identifiziert und definiert (Abschnitt 6.1). Danach wird die Aktualisierung von Verhaltens-Modellen beschrieben (Abschnitt 6.2). Eine objektorientierte Realisierung der Verhaltens-Vorhersage und der Aktualisierung von Verhaltens-Modellen wird vorgestellt (Abschnitt 6.3). Schließlich wird eine Zusammenfassung des eingeführten fallbasierten Verhaltens-Modells $\mathfrak{M}$ gegeben und dessen Abbildung $\mathfrak{f}$ konkretisiert (Abschnitt 6.4).

6.1 Vorhersage von Verhalten

Für eine Entscheidungs-Situation soll das Verhalten der IP vorhergesagt werden. Die Vorhersage gliedert sich in mehrere Teilprozesse. Zuerst gilt es zu entscheiden, ob es sich bei einer Situation um eine Entscheidungs-Situation handelt (Abschnitt 6.1.1). Für eine Entscheidungs-Situationen müssen alle potentiellen Auslöser bestimmt werden (Abschnitt 6.1.2). Basierend auf jedem potentiellen Auslöser wird eine Suche in der Fallbasis des Verhaltens-Modells nach Fällen mit ähnlichen Auslösern durchgeführt und die besten Fälle ausgewählt (Abschnitt 6.1.3). Danach muß aus den ausgewählten Fällen aller Auslöser, der Fall bestimmt werden, der sich am besten für die Vorhersage eignet (Abschnitt 6.1.4). Das Verhalten dieses Falles wird an die aktuelle Entscheidungs-Situation angepaßt und als Ergebnis der Verhaltens-Vorhersage verwendet (Abschnitt 6.1.5). Schließlich werden diese fünf Teilprozesse zum Prozess der Verhaltens-Vorhersage zusammengefasst (Abschnitt 6.1.6).

6.1.1 Erkennung von Entscheidungs-Situationen

In diesem Abschnitt wird untersucht, in welchen Situationen das Verhalten der IP vorhergesagt werden soll. Infolge der Beschränkung auf die ballorientierten Verhaltens-Muster Paß, Torschuß und Dribbeln werden nur Verhalten betrachtet, an deren Start genau ein Spieler den Ball kontrolliert. Demzufolge kommen als Entscheidungs-Situationen nur Situationen in Frage, in denen genau ein Spieler den Ball kontrolliert. Dieser Spieler muß dem modellierten Team angehören.

Es sollten aber nicht alle Situationen, bei denen genau ein Spieler des modellierten Teams den Ball kontrolliert, als Entscheidungs-Situationen betrachtet werden. Wenn ein Spieler über einen längeren Zeitraum die alleinige Kontrolle über den Ball hat, ist eine Vorhersage des Verhaltens für jede Situation nicht sinnvoll, da aufeinanderfolgende Situation nahezu identisch sind und unnötige Vorhersagen vermieden werden sollen.

Nach einer bestimmten Zeit der Ballkontrolle durch einen Spieler, muß die Vorhersage dennoch wiederholt werden. Es stellt sich die Frage, mit welcher Frequenz Vorhersagen sinnvoll sind. Eine längere Ballkontrolle eines Spielers entspricht dem Verhaltens-Muster Dribbeln. Es bietet sich daher an, den Abstand zweier Entscheidungs-Situationen von der minimalen Dribbeldauer (4 Zeittakte) abhängig zu machen.

Demnach wird ein Mindestabstand von 4 Takten zwischen zwei Entscheidungs-Situationen festgelegt, wenn seit der letzten Vorhersage noch kein neues Verhalten beobachtet werden konnte.

6.1.2 Identifizierung von Auslösern

Für eine Entscheidungs-Situation müssen alle potentiellen Auslöser aller betrachteten Verhaltens-Muster bestimmt werden. Dies geschieht mit der Funktion identify_S (vgl. Formel 5.4 auf Seite 111). Diese Funktion wird nacheinander mit allen betrachteten Auslöser-Mustern ausgeführt um alle Auslöser zu identifizieren. Für das Auslöser-Muster Paß werden bis zu 10 Auslöser, für das Auslöser-Muster Torschuß maximal 1 Auslöser und für das Auslöser-Muster Dribbeln genau 6 Auslöser erzeugt (vgl. Abschnitt 5.2.2 auf Seite 110).

6.1.3 Ermittlung von Fällen mit ähnlichen Auslösern

Für jeden identifizierten Auslöser wird in der Fallbasis nach Fällen gesucht, die einen ähnlichen Auslöser haben. Anhand dieser Ähnlichkeit werden die Fälle bewertet. Der für die Ähnlichkeitsbestimmung verwendete Auslöser wird für eine spätere Anpassung des Falles benötigt. Daher wird er als Bestandteil des bewerteten Falles aufgenommen. Die besten der bewerteten Fälle werden ausgewählt und stehen für weitere Betrachtungen zur Verfügung.

Im folgenden wird die Menge für bewertete Fälle eingeführt und die Funktion zur Suche und Auswahl der für einen Auslöser relevanten Fälle angegeben.

`ratedCases` bezeichnet die Menge von bewerteten Fällen. Ein bewerteter Fall besteht aus einem Auslöser $T_* \in$ `triggers`, einem Fall $C \in$ `cases` und

einer reellen Zahl $\sigma \in \mathbb{R}_{[0,1]}$, die die Ähnlichkeit zwischen dem Auslöser T_* und dem Auslöser des Falles C angibt.

$$\texttt{ratedCases} \subset \texttt{triggers} \times \texttt{cases} \times \mathbb{R}_{[0,1]} \tag{6.1}$$

retrieve ist die Funktion, die auf Basis einer Gewichtesequenz für Auslöser $W \in \texttt{weights}_\texttt{T}$ und einer Fallbasis $caseBase \subset \texttt{cases}$ für einen Auslöser $T_* \in \texttt{triggers}$ die relevanten bewerteten Fälle $\{RC_1, \ldots, RC_n\} \subset \texttt{ratedCases}$ ermittelt.

$$\begin{aligned} \text{retrieve} &: \texttt{weights}_\texttt{T} \times 2^{\texttt{cases}} \times \texttt{triggers} \rightarrow 2^{\texttt{ratedCases}} \\ \text{retrieve}(W, caseBase, T_*) &= \{RC_1, \ldots, RC_n\} \quad \text{mit} \\ RC_i &= (T_*, (T_i, BE_i), \sigma_i),\ \sigma_i = \text{similarity}_\text{T}\,(T_*, T_i) \end{aligned} \tag{6.2}$$

Als Ergebnis liefert die Funktion retrieve die bewerteten Fälle, die die höchste Bewertung σ_i haben.

Naiv betrachtet, muß der Auslöser jedes Falles $C \in caseBase$ mit dem Auslöser T_* auf Basis der Funktion $\text{similarity}_\text{T}$ verglichen werden. Da die Ähnlichkeit zwischen Auslösern verschiedener Auslöser-Muster mit Null definiert ist (vgl. Abschnitt 5.5.1 auf Seite 117), entfallen Vergleiche zwischen Auslösern verschiedener Muster.

Prinzipiell sind alle Ähnlichkeitsvergleiche zwischen zwei beliebigen Werten, deren Resultat Null ergibt, überflüssig. Eine solche Reduzierung der Ähnlichkeitsvergleiche ist bei der Verwendung von *Case Retrieval Nets* [36] möglich. Damit kann die Laufzeit der Funktion retrieve stark reduziert werden.

6.1.4 Auswahl eines Auslöser-Verhaltens-Falles

Aus einer Menge von bewerteten Fällen soll der Fall ausgewählt werden, der sich am besten für eine Vorhersage eignet. Zur Ermittlung der besten bewerteten Fälle wurden die primären Attribute des Auslösers der Entscheidungs-Situation verwendet. Bei der Auswahl eines Falles können u.a. die sekundären Attribute genutzt werden.

select ist die Funktion, die aus einer Menge von bewerteten Fällen den Fall auswählt, der sich am besten für die Vorhersage eignet.

$$\begin{aligned} \text{select} &: 2^{\texttt{ratedCases}} \rightarrow \texttt{ratedCases} \\ \text{select}(goodCases) &= bestCase \quad \text{mit } bestCase \in goodCases \end{aligned} \tag{6.3}$$

Prinzipiell können zwei verschiedene Strategien zur Auswahl von Fällen unterschieden werden.

Präferenzen Bei der Verwendung von Präferenzen werden bestimmte Kriterien ausgewählt und die Fälle bevorzugt, die viele dieser Kriterien erfüllen.

Argumentation Bei der Verwendung von Argumentation werden die Fälle miteinander verglichen um den nützlichsten Fall zu bestimmen.

Im Rahmen dieser Arbeit werden zur Auswahl eines Auslöser-Verhaltens-Falles Präferenzen verwendet. Ein Ansatz zur Verwendung von Argumentation wird im Ausblick der Arbeit in Abschnitt 8.3 auf Seite 176 vorgestellt.

Bei der Wahl von Präferenzen gibt es viele Möglichkeiten, im wesentlichen können die sekundären und primären Attribute von Auslöser und Verhalten verwendet werden, um spezielle Fälle bevorzugt auszuwählen. Im Rahmen dieser Arbeit werden Fälle bevorzugt, wenn

- sie den gleichen Spielzustand wie der Auslöser aufweisen,
- sie das gleiche gegnerische Team wie der Auslöser aufweisen,
- sie den gleichen Kicker wie der Auslöser aufweisen,
- ihr Verhalten erfolgreich ist und/oder
- ihr Verhalten ein mittellanger Paß[1] ist.

Bei den ersten drei Kriterien wird jeweils ein sekundäres Attribut des Auslösers des Falles mit dem Auslöser der Entscheidungs-Situation verglichen. Bei dem vierten Kriterium wird ein primäres Attribut des Verhaltens des Falles genutzt und beim fünften Kriterium wird ein primäres Attribut des Auslösers des Falles betrachtet.

Der Grad der Bevorzugung wird durch einen Bonus-Wert für jedes erfüllte Kriterium realisiert. Es wird der Fall ausgewählt, dessen Summe aus Ähnlichkeit und Bonus-Wert maximal ist.

Die Bonus-Werte werden mit dem Verfahren des *Cross Validation* (vgl. Abschnitt 5.5.4.1 auf Seite 120) ermittelt, welches unabhängig von der Bestimmung der Gewichtesequenz für die Auslöser durchgeführt wird.

6.1.5 Anpassung des Verhaltens

Als Ausgangsbasis für die Anpassung steht der durch die Funktion select ausgewählte bewertete Fall $bestCase = (T_*, (T, BE), \sigma) \in \texttt{ratedCases}$ zur Verfügung. Obwohl die Verhaltens-Essenz BE bereits als Lösung verwendet werden kann, können Anpassungen an den verwendeten Auslöser T_* vorteilhaft sein. Für eine Konstruktion einer angepaßten Verhaltens-Essenz BE_* müssen die Unterschiede zwischen den Auslösern T_* und T identifiziert und entsprechend behandelt werden.

adapt ist die Funktion zur Anpassung eines Falles $(T, BE) \in \texttt{cases}$ an einen Auslöser $T_* \in \texttt{triggers}$. Dabei wird der Auslöser T_* mit dem Auslöser des Falles T verglichen. Auf Basis dieses Vergleichs und der Verhaltens-Essenz BE wird eine modifizierte Verhaltens-Essenz BE_* ermittelt.

$$\begin{aligned} &\text{adapt} : \texttt{ratedCases} \rightarrow \texttt{behaviors}_E \\ &\text{adapt}(bestCase) = BE_* \end{aligned} \tag{6.4}$$

[1] Entfernung von Kicker zum Empfänger ist größer als 8 und kleiner als 30 Meter

Prinzipiell können die Werte aller primären und sekundären Attribute der Verhaltens-Essenz *BE* angepaßt werden. Im Rahmen der Arbeit werden

- die Länge des Ballgeschwindigkeitsvektors,
- der Winkel des Ballgeschwindigkeitsvektors und
- die Dauer des Verhaltens

angepaßt.

Beispielhaft soll mit Hilfe der Abbildung 6.1 die Anpassung des Winkels des Ballgeschwindigkeitsvektors erläutert werden. Es sind die zur Anpassung verwendeten primären Attribute der Auslöser T_* und T, als auch der Verhaltens-Essenz *BE* des Verhaltens-Muster Paß dargestellt. Bei den verwendeten Attributen handelt es sich um

- den Vektor vom Kicker zum Empfänger $\overrightarrow{kp}$ des Auslösers T,
- den Vektor vom Kicker zum Empfänger $\overrightarrow{k_*p_*}$ des Auslösers T_* und
- den Winkel des Ballgeschwindigkeitsvektors $s.angle$ des Verhaltens *BE*.

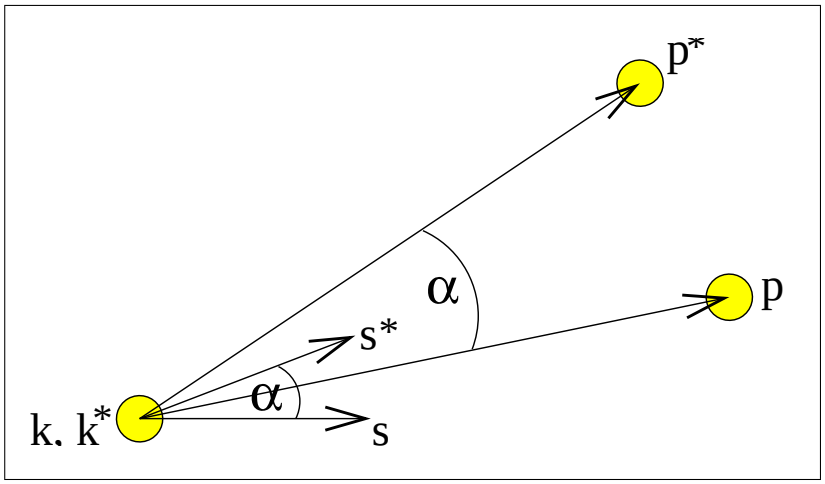


Abbildung 6.1: Anpassung des Winkels des Ballgeschwindigkeitsvektors

Zur Konstruktion des Winkels vom Ballgeschwindigkeitsvektor $s_*.angle$ für die Verhaltens-Essenz BE_* wird der Differenz-Winkel α zwischen den Vektoren $\overrightarrow{kp}$ und $\overrightarrow{k_*p_*}$ gebildet, $\alpha = \overrightarrow{k_*p_*}.angle - \overrightarrow{kp}.angle$ und zum Winkel des Ballgeschwindigkeitsvektors $s.angle$ addiert, $s_*.angle = s.angle + \alpha$.

Besteht Interesse an der Vorhersage des Zieles (Empfänger, Position, oder Richtung) eines Verhaltens, ist das Ziel des Auslösers der Entscheidungs-Situation T_* zu verwenden.

6.1.6 Zusammenfassung der Verhaltens-Vorhersage

Mit den in den letzten Abschnitten eingeführten Funktionen lässt sich die Funktion zur Vorhersage von Verhalten definieren.

predict ist die Verhaltens-Vorhersage Funktion, die auf Basis einer Gewichtesequenz für Auslöser $W \in \texttt{weights}_\texttt{T}$ und einer Fallbasis $caseBase \subset \texttt{cases}$ für eine Situation $S \in \texttt{situations}$ das Verhalten $BE_* \in \texttt{behaviors}_E$ der Interaktionspartner vorhersagt.

$$\begin{aligned} &\text{predict} : \texttt{weights}_\texttt{T} \times 2^{\texttt{cases}} \times \texttt{situations} \rightarrow \texttt{behaviors}_E \\ &\text{predict}(W, caseBase, S) = BE_* \quad \text{mit} \end{aligned} \tag{6.5}$$

$$BE_* = \text{adapt}(bestCase) \qquad \text{aus (6.4)}$$

$$bestCase = \text{select}(goodCases) \qquad \text{aus (6.3)}$$

$$goodCases = \bigcup_{T \in allTriggers} \text{retrieve}(W, caseBase, T) \qquad \text{aus (6.2)}$$

$$allTriggers = \bigcup_{TP \in \texttt{triggerPatterns}} \text{identify}_\text{S}(S, TP) \qquad \text{aus (5.4)}$$

In Abbildung 6.2 ist die Vorhersage eines Verhaltens, basierend auf einer Situation, graphisch dargestellt. Für die Situation S werden alle Auslöser bzgl. der betrachteten Verhaltens-Muster bestimmt. Für jeden dieser Auslöser werden mit der Funktion retrieve die bewerteten Fälle ermittelt, die den jeweiligen Auslöser am ähnlichsten sind. Aus der Vereinigung dieser bewerteten Fälle wird der vielversprechendste mit der Funktion select ausgewählt. Auf Basis dieses bewerteten Falles wird eine Lösung konstruiert, indem die Verhaltens-Essenz des Falles an den Auslöser der Entscheidungs-Situation angepaßt wird.

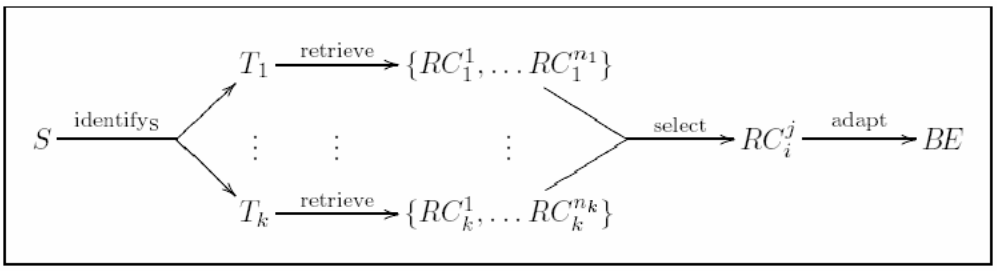


Abbildung 6.2: Vorhersage eines Verhaltens

Die Komplexität der Verhaltens-Vorhersage $o(\text{predict})$ hängt hauptsächlich von der Komplexität der Funktion $o(\text{retrieve})$ und von der Anzahl k der ermittelten Auslöser ab[2]. Die anderen Funktionen haben keinen merklichen Einfluß auf die Laufzeit der Verhaltens-Vorhersage. Damit ergibt sich für die Komplexität der Verhaltens-Vorhersage $o(\text{predict}) = k * o(\text{retrieve})$.

Wie schon erwähnt, werden weit entfernte Spieler und ein weit entferntes Tor nicht als potentielle Ziele bei der Bestimmung von Auslösern betrachtet. Dadurch kann die Anzahl der Auslöser merklich veringert werden. Im ungünstigsten Fall werden aber dennoch 10 Auslöser für den Paß, 1 Auslöser für den Torschuß und 6 Auslöser für das Dribbeln ermittelt.

[2]Die Anzahl der ausgewählten bewerteten Fälle ist sehr klein. Damit ist die Funktion select für die Komplexität der Verhaltens-Vorhersage nicht relevant.

6.2 Aktualisierung von Verhaltens-Modellen

Neben der Generierung von Verhalten-Modellen ist deren Aktualisierung wichtig, da nicht davon ausgegangen werden kann, daß das Modell das Verhalten der IP für alle möglichen Situationen korrekt beschreibt. Es gibt mehrere Gründe, die eine Aktualisierung von Verhaltens-Modellen motivieren. So könnte es sein, daß das Verhalten der IP noch nicht für eine repräsentative Menge von Situationen beobachtet werden konnte. Es ist auch denkbar, daß Fälle mit fehlerhaftem Verhalten der IP in die Fallbasis eingefügt wurden. Schließlich ist es vorstellbar, daß die IP ihre Präferenzen bei der Verhaltensauswahl mit der Zeit ändern.

Ein fallbasiertes Verhaltens-Modell ist für Aktualisierungen besonders gut geeignet, da Änderungen in der Fallbasis durch Aufnahme, Löschung oder Modifizierung eines Falles sogar zur Laufzeit durchführbar sind. Damit ist eine sofortige Anpassung an Verhaltensänderungen der IP möglich. Neben der Aktualisierung der Fallbasis eines Verhaltens-Modells kann auch eine Anpassung der Ähnlichkeitsfunktion oder anderer Teilprozesse vorgenommen werden. Funktionen können dabei durch eine Änderung von verwendeten Parametern angepaßt werden.

Die Aktualisierung von Verhaltens-Modellen wird in eine Aktualisierung der Fallbasis (Abschnitt 6.2.1) und in eine Aktualisierung von verwendeten Parametern (Abschnitt 6.2.2) unterteilt.

6.2.1 Aktualisierung der Fallbasis

Die Aktualiserung der Fallbasis setzt sich aus der Vorhersage und Beobachtung von Verhalten und aus einer eventuellen Integration eines Falles in die Fallbasis zusammen. Ein neuer Fall wird in die Fallbasis aufgenommen, wenn sich das für eine Entscheidungs-Situation vorhergesagte Verhalten von dem entsprechenden beobachteten Verhalten signifikant unterscheidet.

Für eine genaue Erklärung des Prozesses der Aktualisierung der Fallbasis werden einige Bezeichner für die daran beteiligen Auslöser, Verhalten, Verhaltens-Essenzen und Fälle eingeführt. Sei

- B_t^R ein mit der Funktion recognize (Formel 4.11) beobachtetes Verhalten mit dem Startzeitpunkt t,
- T_t^R der mit der Funktion identify_T (Formel 5.5) aus der Situation S_t und dem Verhalten B_t^R bestimmte Auslöser
- BE_t^R die mit der Funktion identify_B (Formel 5.9) aus B_t^R bestimmte Verhaltens-Essenz,
- C_t^R der mit der Funktion generateCase (Formel 5.11) aus dem Auslöser T_t^R und der Verhaltens-Essenz BE_t^R bestimmte Fall und
- BE_t^P die für die Entscheidungs-Situation S_t mit der Funktion predict (Formel 6.5) vorhergesagte Verhaltens-Essenz.

Die Aktualiserung besteht aus einem Ähnlichkeitsvergleich zwischen der vorhergesagten Verhaltens-Essenz BE_t^P und der beobachteten Verhaltens-Essenz BE_t^R. Wenn das Ergebnis dieses Vergleiches $\text{similarity}_\text{B}\,(BE_t^P, BE_t^R)$ kleiner als ein festgelegter Wert σ_limit ist (die Verhaltens-Essenzen sich signifikant unterscheiden), wird der Fall C_t^R in die Fallbasis aufgenommen. Eine Löschung oder Modifizierung von Fällen wird nicht durchgeführt.

reviseModel ist die Funktion zur Aktualisierung der Fallbasis eines Verhaltens-Modells. Auf der Basis einer vorhergesagten Verhaltens-Essenz und eines beoachteten Falles C wird entweder die Fallbasis um den Fall C erweitert oder bleibt unverändert.

$$\begin{aligned} &\text{reviseModel} : 2^{\texttt{cases}} \times \texttt{behaviors}_E \times \texttt{cases} \rightarrow 2^{\texttt{cases}} \\ &\text{reviseModel}\big(caseBase, BE^P, (T^R, BE^R)\big) := \\ &\begin{cases} caseBase \cup \{(T^R, BE^R)\} & \text{wenn } \text{similarity}_{\text{B}}\,(BE^P, BE^R) < \sigma_{\text{limit}} \\ caseBase & \text{sonst} \end{cases} \end{aligned} \tag{6.6}$$

Der Wert für σ_{limit} sollte dabei empirisch bestimmt werden.

Damit wird ein neuer Fall in die Fallbasis aufgenommen, wenn die Vorhersage fehlerhaft war. Eine Erweiterung der Fallbasis bietet sich auch an, wenn diese bzgl. einer Entscheidungs-Situation S_t nicht repräsentativ ist. Dies ist der Fall, wenn für den mit der Funktion select (Formel 6.3) ausgewählten bewerteten Fall $RC_t = (T_*, (T, BE), \sigma) \in \texttt{ratedCases}$, die Ähnlichkeit σ zwischen den Auslösern T_* und T nicht groß genug ist.

6.2.2 Aktualisierungen von Parametern

Die Aktualisierung des Ähnlichkeitsmaßes wird durch eine Neubestimmung der Gewichtesequenz aller Auslöser-Muster (siehe Abschnitt 5.5.4 auf Seite 120) durchgeführt. Aufgrund der hohen Komplexität der Ermittlung einer optimalen Gewichtesequenz (vgl. Abschnitt 5.5.4.2 auf Seite 124 und Abschnitt 5.5.4.3 auf Seite 124) ist die Aktualisierung des Ähnlichkeitsmaßes nicht während des Spieles, sondern erst im nachhinein möglich. Selbst dann ist ein immenser Zeitaufwand nötig, so daß eine Aktualisierung des Ähnlichkeitsmaßes nicht nach jedem Spiel sinnvoll ist.

Die Aktualisierung der Bonus-Werte (vgl. Abschnitt 6.1.4 auf Seite 134) für die Auswahl eines bewerteten Falles mit der Funktion select wird auch mit dem Verfahren des *Cross Validation* durchgeführt. Damit ist die Bestimmung der Bonus-Werte ebenfalls sehr zeitaufwendig. Im Gegensatz zur Aktualisierung des Ähnlichkeitsmaßes werden die Bonus-Werte nicht neu bestimmt, sondern innerhalb festgelegter Grenzen verändert, um eine optimierte Bonus-Werte-Sequenz zu ermitteln.

6.3 Objektorientierte Realisierung

Für die in diesem Kapitel eingeführten Mengen und Funktionen wird eine objektorientierte Realisierung angegeben. Die im Abschnitt 5.6.4 auf Seite 129 vorgestellte Klasse **Teammodel** wird um Methoden zur Vorhersage von Verhalten und zur Aktualisierung von Verhaltens-Modellen erweitert. Die Umsetzung der eingeführten Teilprozesse der Verhaltens-Vorhersage in Methoden erfolgt dabei nahezu eins zu eins.

Bei der Anwendung von Verhaltens-Modellen ist mit der Menge der bewerteten Fälle `ratedCases` eine neue Datenstruktur dazugekommen. Die Menge wird

durch die Klasse **RatedCases** realisiert und ist in Abbildung 6.3 dargestellt. Objekte der Klasse **RatedCases** werden bei der Vorhersage von Verhalten erzeugt und weiterverarbeitet.

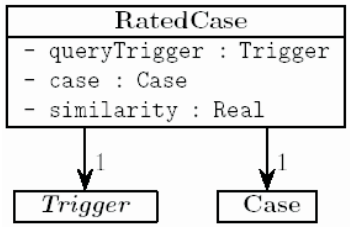


Abbildung 6.3: Klassendiagramm für die bewerteten Fälle

Die für die Anwendung von Verhaltens-Modellen wichtigsten Bestandteile der Klasse **Teammodel** sind in Abbildung 6.4 wiedergegeben.

Dabei ist zu beachten, daß die Fallbasis *caseBase* in die drei Teile *passCases*, *scoreGoalCases* und *dribbleCases* zerlegt wurde, die jeweils nur die Fälle eines Verhaltens-Musters enthalten. Mit dieser Unterteilung kann die Ermittlung von ähnlichen Fällen bzgl. eines Auslösers auf den relevanten Teil aller Fälle begrenzt werden (vgl. Abschnitt 6.1.3 auf Seite 133).

Mit der Liste *bestCases* werden die relevanten bewerteten Fälle festgehalten, die die höchste Bewertung σ haben. Diese Liste wird für Zwischenergebnisse bei der Vorhersage verwendet und hat eine feste Anzahl von Elementen[3].

Da die Aktualisierung der Fallbasis erst nach der Beobachtung eines Verhaltens vorgenommen werden kann, müssen die vorhergesagten Verhalten gespeichert werden. Dabei ist es möglich, daß mehrere Vorhersagen gemacht werden, bevor tatsächlich ein Verhalten erkannt wird. Die nach dem letzten beobachteten Verhalten, vorhergesagten Verhaltens-Essenzen werden in der Menge *predictedBehaviors* gespeichert.

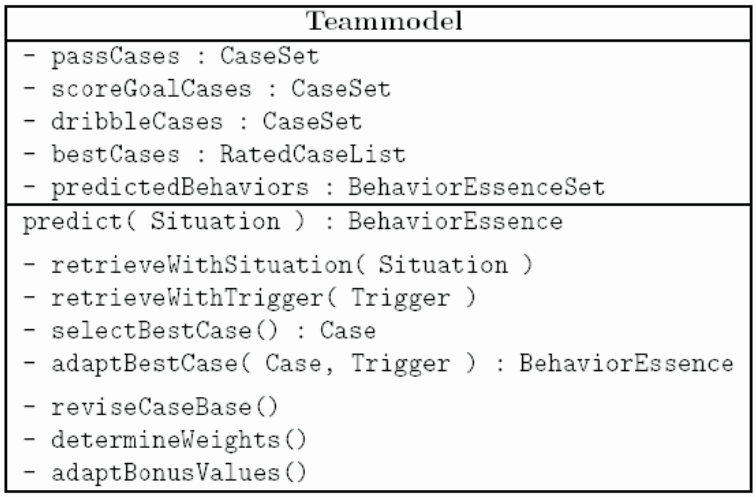


Abbildung 6.4: Erweiterung des Klassendiagramms eines Teammodels

[3]Es werden die 10 bestbewerteten Fälle aller Auslöser gesammelt.

Der Prozess der **Verhaltens-Vorhersage** ist in der Methode *predict* realisiert. Für eine Entscheidungs-Situation bestimmt diese Methode eine Verhaltens-Essenz. Dafür werden die Methoden *retrieveWithSituation*, *retrieveWithTrigger*, *selectBestCase* und *adaptBestCase* verwendet.

Die Methode *retrieveWithSituation* initialisiert zuerst die Liste der relevanten bewerteten Fälle *bestCases*. Danach werden alle Auslöser für die Entscheidungs-Situation bestimmt und für jeden Auslöser die Methode *retrieveWithTrigger* ausgeführt.

Mit der Methode *retrieveWithTrigger* werden die für den übergegeben Auslöser relevanten Fälle in der entsprechenden Fallbasis gesucht. Für jeden Fall der Fallbasis wird die Ähnlichkeit mit dem Auslöser bestimmt und ein bewerteter Fall generiert. Dieser wird anhand der Ähnlichkeit σ in die Liste *bestCases* einsortiert. Da die Initialisierung dieser Liste bereits in der Methode *retrieveWithSituation* geschieht, teilen sich alle Auslöser die Liste *bestCases*. Ein bewerteter Fall gilt dabei vorerst als relevant bzgl. eines Auslösers, wenn dessen Bewertung σ größer als die schlechteste Bewertung der Fälle in der Liste *bestCases* ist oder sich noch nicht 10 Fälle in der Liste befinden. Bei der Aufnahme eines neuen Falles in eine bereits volle Liste wird der Fall mit der schlechtesten Bewertung aus der Liste entfernt.

Nach der vollständigen Abarbeitung der Methode *retrieveWithSituation* enthält damit die Liste *bestCases* die n bestbewerteten Fälle aller untersuchten Auslöser. Die Methode *selectBestCase* bestimmt für alle Fälle dieser Liste die Bonus-Werte und wählt den bewerteten Fall aus, dessen Summe aus Ähnlichkeit σ und Bonus-Werten maximal ist. Weisen mehrere bewertete Fälle die gleiche maximale Summe auf, wird von den maximal bewerteten Fällen ein zufälliger ausgewählt.

Dieser ausgewählte Fall wird von der Methode *adaptBestCase* verwendet um eine angepaßte Verhaltens-Essenz zu bestimmen. Diese Verhaltens-Essenz ist das Endergebnis der Verhaltens-Vorhersage durch die Methode *predict* und wird für Revisionsbetrachtungen in die Menge *predictedBehaviors* aufgenommen.

Die **Aktualisierung des Verhaltens-Modells** ist durch die Methoden *reviseCaseBase*, *determineWeights* und *adaptBonusValues* realisiert.

Die Methode *reviseCaseBase* wird ausgeführt, wenn ein Verhalten der IP beobachtet wurde. Dann wird jedes vorhergesagte Verhalten aus der Menge *predictedBehaviors* (normalerweise ist dies genau ein Verhalten) mit dem erkannten Verhalten auf Basis des Ähnlichkeitsmaßes verglichen. Für jedes Verhalten dessen Ähnlichkeit mit dem vorhergesagten Verhalten zu gering ist (kleiner als ein Mindestwert σ), wird ein neuer Fall generiert und in die Fallbasis aufgenommen. Schließlich wird die Menge *predictedBehaviors* neu initialisiert.

Die Methoden *determineWeights* und *adaptBonusValues* passen die Gewichtesequenzen der Auslöser bzw. die Bonus-Werte zur Auswahl eines Falles an. Dabei wird das Verfahren des *Cross Validation* genutzt. Während die Methode *determineWeights* die beste Gewichtesequenz ermittelt (dabei wird die bisherige Gewichtesequenz nicht betrachtet), werden bei der Suche nach optimierten Bonus-Werten mit der Methode *adaptBonusValues* die bisher verwendeten Bonus-Werte in fest definierten Intervallen verändert.

Die Verwendung und die Parameter vieler Methoden sind konfigurierbar, so

kann z.B. die Aktualisierung des Verhaltens-Modells für alle drei Methoden an- und abgestellt werden.

6.4 Resümee: Fallbasiertes Verhaltens-Modell

Die Bestandteile des verwendeten fallbasierten Verhaltens-Modells sind damit vollständig defininiert. Das Fallbasierte System besteht aus einer Fallbasis, einem Ähnlichkeitsmaß, einer Vorhersage-Funktion und einer Aktualisierungs-Funktion.

Die Fallbasis *caseBase* besteht aus einer Menge von Auslöser-Verhaltens-Fällen. Mit Methoden zur Generierung (generateCase) und Aufnahme (integrateCase) eines Falles kann die Fallbasis verändert werden.

Das Ähnlichkeitsmaß ist durch die Funktionen $\text{similarity}_\text{T}$ und $\text{similarity}_\text{B}$ gegeben, die die Ähnlichkeit basierend auf Auslöser bzw. Verhaltens-Essenzen ermitteln. Als variable Größe dieser Funktionen wird die Gewichtesequenz der Auslöser-Muster $\texttt{weights}_\text{T}$ verwendet.

Die Vorhersage-Funktion predict basiert auf den vier Funktionen identify_S, retrieve, select und adapt. Für eine Situation $S \in \texttt{situations}$ wird das Verhalten von Interaktionspartnern in Form einer Verhaltens-Essenz $BE \in \texttt{behaviors}_E$ vorhergesagt.

Die Aktualisierungs-Funktion reviseModel vergleicht vorhergesagtes Verhalten mit tatsächlich beobachtetem Verhalten. Basierend auf dem Ergebnis des Vergleichs wird ein neuer Fall in die Fallbasis aufgenommen oder nicht. Zur Aktualisierung des Verhaltens-Modells gehören auch die Anpassungen der Gewichtesequenz der Auslöser und die Anpassung der Bonus-Werte zur Auswahl eines Falles.

Die Funktion $\mathfrak{f}$ des Verhaltens-Modells $\mathfrak{M} = [\texttt{situations}, \texttt{behaviors}_E, \mathfrak{f}]$ (vgl. Abschnitt 3.4 auf Seite 36) ist im wesentlichen durch die Funktionalität der Vorhersage-Funktion predict gegeben, so daß sie wie folgt definiert werden kann:

$$\mathfrak{f}\colon \texttt{situations} \rightarrow \texttt{behaviors}_E$$

$$\mathfrak{f}(S) = \begin{cases} \text{predict}(W, \mathit{caseBase}, S) & S \text{ ist eine Entscheidungs-Situation} \\ \bot & \text{sonst} \end{cases} \tag{6.7}$$

Dabei ist $W \in \texttt{weights}_\text{T}$ die verwendete Gewichtesequenz für Auslöser und $\mathit{caseBase} \subset \texttt{cases}$ die verwendete Fallbasis. Die Gewichtesequenz und die Fallbasis können sich zwischen zwei Aufrufen der Funktion $\mathfrak{f}$ ändern.

Kapitel 7

Evaluierung des Ansatzes

Der vorgestellte Ansatz zur Erkennung und Vorhersage von Verhalten wird in diesem Kapitel evaluiert. Dazu wurde die in den einzelnen Abschnitten beschriebene objektorientierte Realisierung weiter konkretisiert und in der Programmiersprache C++ implementiert. Die Implementation wurde für verschiedene Szenarien getestet. In diesem Kapitel werden diese Szenarien beschrieben und die Resultate der Untersuchungen angegeben und ausgewertet.

Zuerst werden einige allgemeine Bemerkungen zur Implementation und zu den verwendeten Primärdaten gemacht (Abschnitt 7.1). Bei der Evaluierung der Verhaltens-Bestimmung werden im wesentlichen die vom Coach erkannten Verhalten mit den durch einen Beobachter erkannten Verhalten verglichen (Abschnitt 7.2). Die Verhaltens-Vorhersage wird automatisch evaluiert, der Coach vergleicht selbständig die von ihm vorhergesagten Verhalten mit den Verhalten, die er erkannt hat (Abschnitt 7.3). Schließlich werden die Ergebnisse der Erkennung und der Vorhersage von Verhalten zusammengefasst (Abschnitt 7.4).

7.1 Allgemeine Bemerkungen

Die Entwicklung und Evaluierung erfolgte unter *Linux* (Kernel 2.2.13) auf einem Computer mit einem 400 MHz getakteten *AMD-K6-III-3D* Prozessor und 64 MB Arbeitsspeicher. Als Compiler wurde der *gcc* in der Version egcs-2.91.66 verwendet.

Bezüglich der Implementation des Coach-Programmes wird geschildert, inwieweit die in dieser Arbeit vorgestellten Konzepte umgesetzt wurden und welchen Umfang (in Programmzeilen, Dateien und Klassen) die Implementation hat (Abschnitt 7.1.1). Als Primärdaten werden die aufgezeichneten Spiele eines Turniers genutzt. Sie werden vorgestellt und ihre Verwendung wird motiviert (Abschnitt 7.1.2).

7.1.1 Implementation

Neben der Verwendung bereits existierender Funktionalität wurden über 12.600 Zeilen implementiert, die sich auf 106 Dateien bzw. 53 Klassen verteilen. Die realisierte Funktionalität erstreckt sich über mehrere Bereiche, die man in die

Aufbereitung von Primärdaten, in die Verhaltens-Bestimmung und die Verhaltens-Vorhersage unterteilen kann.

Aufbereitung Primärdaten: Zur Aufbereitung der vom Coach verwendeten Primärdaten (siehe Abbildung 3.8 auf Seite 33) wurden bereits existierende Bestandteile des Agenten-Teams *AT Humboldt 2000* [6] und des *SoccerServer* [46] benutzt. Zum Einlesen der Coach-spezifischen Daten, sowohl direkt im Spiel, als auch aus Aufzeichnungen heraus, wurden weitere Schnittstellen und Basisklassen in einem Umfang von 4410 Zeilen implementiert, die sich auf 13 Klassen und 26 Dateien verteilen.

Verhaltens-Bestimmung: Die Verhaltens-Bestimmung wurde für die Verhaltens-Muster Balltransfer, Paß, Dribbeln, Torschuß, Klären, Doppelpaß und „Kampf um alleinige Ballkontrolle“ implementiert. Mit dem Verhaltens-Muster „Kampf um alleinige Ballkontrolle“ werden alle Verhalten repräsentiert, bei deren Start mehrere Spieler die Kontrolle über den Ball haben und deren Ende durch die alleinige Ballkontrolle eines Spielers oder durch einen Wechsel in den Spielzustand Freistoß etc. gegeben ist.

Die Einschränkung auf die genannten Verhaltens-Muster resultiert vor allem aus der sehr guten Vergleichbarkeit der Analyseergebnisse des Coaches mit menschlichen Beobachtungen. So sind Start- und Endzeitpunkte der Verhalten klar erkennbar, und paralleles Auftreten von Verhalten ist selten. Der vorgestellte Formalismus zur Verhaltens-Spezifikation und der Algorithmus zur Verhaltens-Erkennung sind auch dazu geeignet, komplexere und weniger ballzentrierte Verhaltens-Muster zu beschreiben und zu analysieren.

Die Implementation zur Verhaltens-Bestimmung hat einen Umfang von 5210 Zeilen, die sich über 64 Dateien bzw. 32 Klassen erstrecken. Der Hauptanteil der Klassen wird dabei durch die Ereignis-Klassen gebildet.

Verhaltens-Vorhersage: Die Generierung und Anwendung des Verhaltens-Modells wurde für das Verhaltens-Muster Paß implementiert. Die Betrachtung des Verhaltens-Musters Paß ist hierbei ausreichend, da dieses Verhaltens-Muster bereits die wichtigsten Eigenschaften aller in Kapitel 5 untersuchten Verhaltens-Muster vereint. So existieren beim Paß bereits mehrere potentielle Ziele (Empfänger). Desweiteren hängt der Erfolg eines Passes nicht nur von den Verhalten des Kickers und des Empfängers, sondern auch von den Handlungen der gegnerischen Spieler ab.

In der aktuellen Implementation wird jeder Paß-Auslöser mit jedem Fall der Fallbasis verglichen. Durch eine Reduzierung der Ähnlichkeitsvergleiche (effizienteres Retrieval) in der Fallbasis (vgl. Abschnitt 6.1.3 auf Seite 132) ist damit noch erhebliches Potential für Laufzeitverbesserungen der Verhaltens-Vorhersage vorhanden.

Die Implementation der Generierung und Anwendung von Verhaltens-Modellen besteht aus 3010 Zeilen, die sich auf 16 Dateien bzw. 8 Klassen verteilen.

7.1.2 Primärdaten

Als Primärdaten werden die Aufzeichnungen der Spiele der *German Open 01*[1] verwendet. Die Spiele der *German Open 01* wurden aufgrund der folgenden Eigenschaften ausgewählt:

- es sind realistische Spiele, wobei die Teams des Turniers unterschiedlichste Spielstärken aufweisen; bei dem Turnier waren sowohl die damalig weltweit stärksten Teams[2], als auch schwächere Teams vertreten
- für diese Spiele existieren bereits Vergleichsdaten bzgl. der Verhaltens-Bestimmung (Diplomarbeit von Müller [43]), so daß die Ergebnisse der Verhaltens-Bestimmung miteinander verglichen werden können und
- es handelt sich um einen überschaubaren Datensatz von Spielen der gut strukturiert ist

An den *German Open 01* haben 12 Teams[3] teilgenommen, die erst in zwei Vorrunden (die jeweils 4 besten Teams jeder Gruppe haben sich qualifiziert) und dann im Doppel-KO-System gegeneinander angetreten sind. Damit stehen 44 Spiele zur Verfügung, bei denen 4 Teams genau 5 Spiele und die anderen mindestens 7 Spiele gespielt haben.

Für die einzelnen Evaluierungen werden nicht immer alle 44 Spiele verwendet, sondern eine relevante Teilmenge der Spiele ausgewählt. Während der automatisierbare Teil der Evaluierung der Verhaltens-Bestimmung anhand aller Spiele erfolgt, werden beim manuellen Teil der Evaluierung der Verhaltens-Bestimmung die 10 Spiele verwendet, die bereits in [43] untersucht wurden. Bei der Verhaltens-Vorhersage werden die Spiele jeweils in Abhängigkeit von der konkreten Evaluierungsaufgabe ausgewählt.

Die Verwendung von Aufzeichnungen hat den Vorteil, daß sich verschiedene Ansätze unter den gleichen Bedingungen testen und vergleichen lassen. Trotz der Beschränkung der Evaluierung auf Aufzeichnungen von Spielen ist es mit der vorliegenden Implementation ebenso möglich, gerade stattfindende Spiele zu analysieren.

7.2 Evaluierung der Verhaltens-Bestimmung

Die Evaluierung der Verhaltens-Bestimmung erfolgt größtenteils manuell, das heißt ein menschlicher Beobachter bestimmt die von den Spieler-Agenten ausgeführten Verhalten und vergleicht diese mit den durch den Coach erkannten Verhalten. Aufgrund dieser Vergleiche werden Werte für **Korrektheit** und **Vollständigkeit**[4] der Verhaltens-Erkennung ermittelt.

[1] Die *German Open 01* fand im Mai 2001 in Paderborn statt.

[2] Die beiden Finalisten der *German Open 01* (FCPortugal und Brainstormers01) wurden bei der 2 Monate später stattfindenen Weltmeisterschaft Zweiter und Dritter.

[3] Die meisten der an diesem Turnier teilnehmenden Mannschaften sind in [3] beschrieben.

[4] Der in diesem Kapitel verwendete Vollständigkeits-Begriff ist nicht zu verwechseln mit dem in Abschnitt 4.4.5 auf Seite 74 verwendeten Vollständigkeits-Begriff. Dort wurde die Vollständigkeit der Verhaltens-Spezifikation anhand der Balltransfermöglichkeiten eines Kickers untersucht, hier wird die Vollständigkeit der Verhaltens-Erkennung anhand der durch einen Beobachter erkannten Verhalten betrachtet.

Ein Teil der Evaluierung läßt sich aber auch automatisieren. So kann automatisch untersucht werden, wie umfassend die Handlungen der IP durch die einzelnen Verhaltens-Muster repräsentiert (abgedeckt) sind. Es läßt sich bestimmen, wie groß der Anteil an den Zeittakten eines Spieles ist, der durch den Coach als eines der beschriebenen Verhaltens-Muster erkannt wird. Dieser Anteil wird im folgenden als **Abdeckung** bezeichnet.

In diesem Abschnitt werden nach Bemerkungen zur Laufzeit der Verhaltens-Erkennung (Abschnitt 7.2.1), automatische Untersuchungen zur Bestimmung der Abdeckung (Abschnitt 7.2.2) und manuelle Untersuchungen zur Ermittlung der Korrektheit und Vollständigkeit (Abschnitt 7.2.3) der Verhaltens-Erkennung betrachtet. Dabei wird jeweils die Versuchsplanung beschrieben und die Ergebnisse der Analysen werden angegeben und ausgewertet.

7.2.1 Laufzeit der Verhaltens-Erkennung

Bei Verwendung des oben angegebenen Testrechners (400 MHz Prozessortakt) benötigt der Coach für die Verhaltens-Erkennung in jedem Simulationstakt maximal eine Millisekunde Rechenzeit. Zusammen mit dem Einlesen und Aufbereiten der Daten sowie der Ausgabe der erkannten Verhaltens-Muster werden für die Analyse eines aufgezeichneten Spieles normaler Länge (6000 Simulationstakte) weniger als 4 Sekunden benötigt.

Damit ist klar, daß die vorliegende Implementation der Verhaltens-Erkennung ohne Einschränkung auch für die Analyse eines gerade stattfindenden Spieles verwendet werden kann – auch bei der gleichzeitigen Analyse von deutlich mehr Verhaltens-Mustern[5]. Bezüglich der Laufzeit kann der Algorithmus zur Verhaltens-Erkennung auch in jedem Spieler-Agenten zum Einsatz kommen. Allerdings stehen den Spielern nur unvollständige und meist sehr ungenaue Daten über ihre Umwelt zur Verfügung. Die verwendeten Verhaltens-Muster sind an diese Bedingungen nicht angepaßt. Selbst unter der Voraussetzung einer entsprechenden Anpassung sind viel ungenauere Analysergebnisse zu erwarten.

7.2.2 Automatische Analysen zur Abdeckung

Versuchsplanung: Wie schon angedeutet, sollen die durch den Coach wahrgenommenen Anteile der einzelnen Verhaltens-Muster an der relevanten Gesamtspielzeit (Spielzeit in der die prinzipielle Möglichkeit der Durchführung eines der implementierten Verhaltens-Muster besteht) bestimmt werden. Insbesondere soll ermittelt werden, in welchem Umfang die Handlungen der IP nicht durch die Verhaltens-Muster Paß, Dribbeln, Torschuß, Klären und „Kampf um alleinige Ballkontrolle“ bzw. nicht durch das allgemeine Verhaltens-Muster Balltransfer erkannt werden.

Für die Untersuchungen zur Abdeckung werden alle 44 Spiele automatisch ausgewertet und jeweils die prozentualen Anteile der einzelnen Verhaltens-Muster pro Spiel bestimmt. Desweiteren wird der prozentuale Anteil der Simulationstakte ermittelt, für die kein Verhaltens-Muster erkannt werden konnte.

[5] Die Laufzeit der Verhaltens-Erkennung ist linear von der Anzahl der Verhaltens-Muster abhängig.

Dazu werden jeweils die Ausschnitte der Spiele betrachtet, in denen die prinzipielle Möglichkeit zur Durchführung eines der oben genannten Verhaltens-Muster besteht. Die Gesamtzeit dieser Ausschnitte in Simulationstakten wird im folgenden mit $playOnTime$ bezeichnet. Die Anzahl der Simulationstakte für die mindestens eines der aufgezählten Verhaltens-Muster ohne bzw. inklusive dem Verhaltens-Muster Balltransfer erkannt wurde, werden mit $behaviorTime$ bzw. $transferTime$ bezeichnet.

Damit ergeben sich zwei Arten von Abdeckungen: $Abdeckung_{Behavior}$ und $Abdeckung_{Transfer}$, die in den Formeln 7.1 und 7.2 definiert sind.

$$Abdeckung_{Behavior} = \frac{behaviorTime}{playOnTime} * 100\% \tag{7.1}$$

$$Abdeckung_{Transfer} = \frac{transferTime}{playOnTime} * 100\% \tag{7.2}$$

Neben der Bestimmung der prozentualen Anteile der einzelnen Verhaltens-Muster wird für alle Beobachtungssequenzen, in denen ein Balltransfer-Verhalten aber kein anderes Verhaltens-Muster erkannt wurde, untersucht, welche der in Abschnitt 4.4.5 auf Seite 74 besprochenen Problemfälle auftreten. Die prozentualen Anteile der Problemfälle werden angegeben.

Ergebnisse: Die Anteile der einzelnen Verhaltens-Muster bzgl. der relevanten Gesamtspielzeit $playOnTime$ sind in Abbildung 7.1 dargestellt. Dabei handelt es sich um die Mittelwerte der jeweiligen Anteile der 44 untersuchten Spiele. Die genauen Werte sind u.a. in Tabelle 7.1 auf der nächsten Seite wiedergegeben.

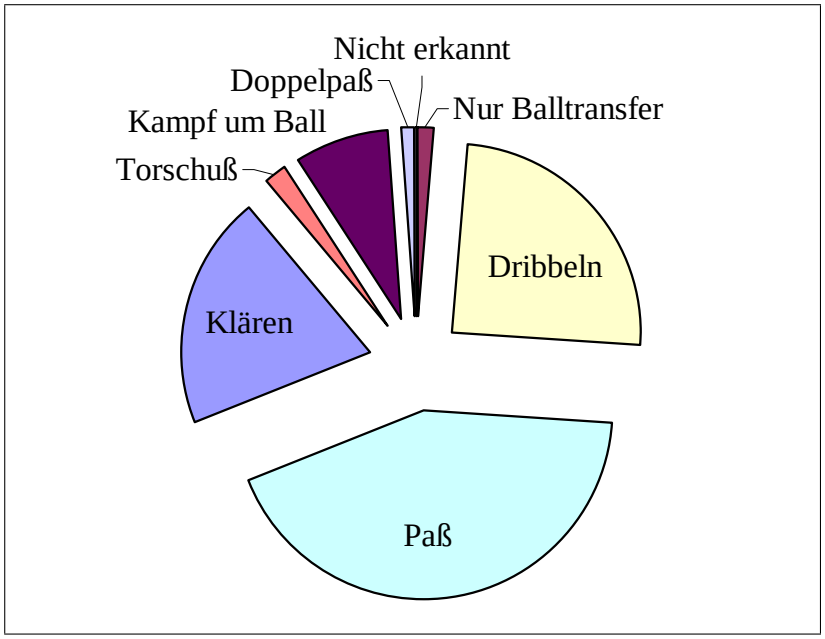


Abbildung 7.1: Durchschnittliche Häufigkeiten der Verhaltens-Muster

Tabelle 7.1 zeigt die prozentualen Anteile der einzelnen Verhaltens-Muster. In der Spalte „Simulationstakte“ sind die Anteile bzgl. der Anzahl der Simulationstakte *playOnTime*, in der Spalte „Verhaltens-Anzahl“ bzgl. der Gesamtanzahl der erkannten Verhalten aufgeführt. Es sind jeweils die Mittel- (avg), Minimal- (min) und Maximalwerte (max) der jeweiligen Anteile in den 44 Spielen angegeben.

In der Zeile „Nicht erkannt“ ist der prozentuale Anteil der Simulationstakte wiedergegeben, für die kein Verhaltens-Muster erkannt werden konnte. Die relevante Gesamtzeit *playOnTime* bzw. die Gesamtanzahl der in dieser Zeit beobachteten Verhalten sind in der Zeile „Gesamtanzahl“ dargestellt. Schließlich sind die Resultate für die beiden Abdeckungen $Abdeckung_{Behavior}$ und $Abdeckung_{Transfer}$ aufgeführt.

Verhaltens-Muster	Simulationstakte			Verhaltens-Anzahl		
	avg	min	max	avg	min	max
Nicht erkannt	0,13	0,00	0,54	—	—	—
Nur Balltransfer	1,41	0,25	3,55	1,87	0,46	4,29
Paß	43,42	32,12	55,74	45,03	32,85	53,97
Dribbeln	24,75	14,65	42,45	28,36	18,41	37,18
Torschuß	1,95	0,25	5,57	2,40	0,49	6,23
Klären	20,22	11,66	28,72	13,00	7,33	19,95
Kampf um Ball	8,20	1,36	13,32	8,79	2,72	14,03
Doppelpaß	1,09	0,00	4,00	0,75	0,00	2,38
Gesamtanzahl	5457	3519	8891	478	257	709
$Abdeckung_{Behavior}$	98,46	96,36	99,68	—	—	—
$Abdeckung_{Transfer}$	99,87	99,46	100,00	—	—	—

Tabelle 7.1: Prozentuale Anteile der einzelnen Verhaltens-Muster

Die nur als Balltransfer erkannten Verhalten wurden mit den sogenannten Problemfällen weiter untergliedert. Bei der Beobachtung eines Balltransfers durch den Coach ermittelt dieser die Art des Problemfalles. Bei der Analyse der 44 Spiele der *German Open 01* konnten im wesentlichen nur die Problemfälle 3 (langsamer Paß) und 5 (langsames Klären) festgestellt werden (zu den Problemfällen vgl. Abschnitt 4.4.5 auf Seite 74). In Tabelle 7.2 sind die Anzahl und die prozentualen Anteile der einzelnen Problemfälle aller 44 Spiele dargestellt.

Problem	Anzahl	Anteil in Prozent
langsamer Paß (3)	257	65,56
langsames Klären (5)	123	31,38
andere	12	3,06

Tabelle 7.2: Anteile der Problemfälle bei den nur als Balltransfer erkannten Verhalten für alle 44 Spiele

Auswertung: Mit den Verhaltens-Anteilen aus Tabelle 7.1 auf der vorherigen Seite ist gezeigt, daß

- der prozentuale Anteil der Simulationstakte für die keines der betrachteten Verhalten erkannt werden konnte sehr gering ist (durchschnittlich 0,13%), damit ist die Abdeckung $Abdeckung_{Transfer}$ mit durchschnittlich 99,87% sehr groß,

- der prozentuale Anteil der Simulationstakte für die nur ein Balltransfer-Verhalten beobachtet werden konnte gering ist (durchschnittlich 1,41%), damit ist die Abdeckung $Abdeckung_{Behavior}$ mit durchschnittlich 98,46% auch noch sehr groß,

- Paß-Verhalten den größten Anteil der beobachteten Verhalten ausmachen (durchschnittlich 43,42% bzgl. der Simulationstakte) und sich damit prinzipiell zur Verhaltens-Vorhersage anbieten.

Wie schon in Abschnitt 4.4.5 auf Seite 74 festgestellt wurde, schließen sich die Verhaltens-Muster Torschuß und Paß im Fehlerfall nicht aus. Desweiteren besteht ein Doppelpaß aus zwei Pässen, die auch einzeln erkannt werden. Damit ist die Summe der Anteile der Verhaltens-Muster im allgemeinen größer als 100%. Bei den Beobachtungssequenzen, in denen kein Verhalten erkannt werden konnte, handelt es sich hauptsächlich um sehr kurze „Dribbel-Manöver“, die aufgrund der Mindestzeit beim Dribbeln vom Coach noch nicht als Dribbel-Verhalten erkannt werden.

7.2.3 Manuelle Analysen zur Korrektheit und Vollständigkeit

Versuchsplanung: In diesem Abschnitt wird die Güte der Verhaltens-Bestimmung untersucht. Da die Spezifikation der Verhaltens-Muster auch auf intuitiven und subjektiven Vorstellungen beruht, wird unter der Güte, das Maß der Übereinstimmung zwischen den durch den Coach und den durch einen menschlichen Beobachter erkannten Verhalten betrachtet. Die Übereinstimmung wird anhand der Korrektheit und der Vollständigkeit der Verhaltens-Erkennung evaluiert.

Korrektheit und Vollständigkeit sind dabei skalare Größen im Intervall $[0, 1]$. Die Güte der Verhaltens-Bestimmung ist um so besser, je höher die Werte für Korrektheit und Vollständigkeit sind. Dabei ist insbesondere die Betrachtung beider Werte wichtig, da die Verbesserung des einen Wertes oftmals eine Verschlechterung des anderen Wertes mit sich bringt. So werden z.B. bei einer sehr speziellen Spezifikation der Verhaltens-Muster nur die wirklich typischen Verhalten erkannt. Dies führt zu einer hohen Korrektheit aber auch zu einem schlechten Wert für die Vollständigkeit. Bei einer zu allgemeinen Spezifikation der Verhaltens-Muster erhöht sich zwar die Vollständigkeit, dafür sinkt aber auch die Korrektheit.

Die Menge der vom Coach erkannten Verhalten wird im folgenden mit V_C und die Menge der vom menschlichen Beobachter erkannten Verhalten mit V_M

bezeichnet. Für die Bestimmung der Korrektheit[6] werden die Verhalten, die durch den Coach und den Menschen erkannt wurden ($V_C \cap V_M$), zu den Verhalten ins Verhältnis gesetzt, die durch den Coach allein erkannt wurden (V_C). Zur Ermittlung der Vollständigkeit werden die durch Coach und Mensch erkannten Verhalten ($V_C \cap V_M$) den Verhalten gegenübergestellt, die durch den menschlichen Beobachter erkannt wurden (V_M). Korrektheit und Vollständigkeit sind in den Formeln 7.3 und 7.4 definiert.

$$Korrektheit = \frac{|V_C \cap V_M|}{|V_C|} = k \tag{7.3}$$

$$Vollständigkeit = \frac{|V_C \cap V_M|}{|V_M|} = v \tag{7.4}$$

Die Erkennung der Verhalten durch den Menschen wird mit Hilfe der Programme *SoccerMonitor* und *Logplayer* durchgeführt, die beide Bestandteile des *SoccerServer*-Paketes sind. Der *SoccerMonitor* stellt den Spielverlauf graphisch aus der Vogelperspektive dar. Der *Logplayer* bietet die Möglichkeit, ein aufgezeichnetes Spiel mit dem *SoccerMonitor* abzuspielen und hat umfangreiche Steuerungsmöglichkeiten (Play, Stop, taktweise vorwärts und rückwärts, direkte Anwahl von Takten).

Für ein Untersuchungsintervall werden alle in diesem Intervall beginnenden Verhalten der Verhaltens-Muster Paß, Dribbeln, Torschuß, Klären und „Kampf um alleinige Ballkontrolle“ ermittelt sowie das alleinige Auftreten von Balltransfers registriert[7]. Für ein erkanntes Verhalten werden dessen Name, Startzeitpunkt, Endzeitpunkt und die beteiligten Spieler notiert. Desweiteren wird festgehalten, ob das Verhalten erfolgreich oder erfolglos war. Damit wird auch ein Teil der Verhaltens-Ausprägungen evaluiert. Weitere Ausprägungen der Verhalten werden nicht notiert.

Beim Vergleich der Beobachtungsergebnisse von Mensch und Coach sollen zwei Verhalten als übereinstimmend (identisch) gelten, wenn deren Namen, beteiligte Spieler und die Ausprägungen übereinstimmen und sich die Werte für den Start- und Endzeitpunkt nicht mehr als einen Simulationstakt unterscheiden.

Da die Beobachtung durch den Menschen subjektiv ist, wird die manuelle Evaluierung der Verhaltens-Bestimmung in 3 Schritten vorgenommen.

1. Zuerst werden die Untersuchungsintervalle vom Menschen beobachtet und die erkannten Verhalten protokolliert.

2. Dann werden die Verhalten durch den Coach bestimmt und ein Vergleich mit den vom Menschen protokollierten Verhalten durchgeführt.

[6] Die verwendeten Definitionen für Korrektheit und Vollständigkeit orientieren sich an den beispielsweise im Bereich des Fallbasierten Schließens gebräuchlichen Maße für *Precision* (Korrektheit) und *Recall* (Vollständigkeit).

[7] Doppelpaß-Verhalten werden nicht ermittelt, da diese sehr selten sind und erhöhte Aufmerksamkeit von Seiten des menschlichen Beobachters erfordern (Beobachter müsste Verhalten parallel beobachten).

3. Für voneinander abweichend erkannte Verhalten findet schließlich eine Überprüfung statt. Es wird untersucht, ob sich der menschliche Beobachter geirrt hat, bzw. die abweichende „Interpretation“ des Coaches dennoch zutreffend ist. Gegebenenfalls wird eine Korrektur der menschlichen Beobachtung vorgenommen.

Somit werden die Werte für die *Vollständigkeit* (v) und die *Korrektheit* (k) zweimal bestimmt, in Schritt 2 und Schritt 3. Eine Unterscheidung dieser Werte wird über Indizes vorgenommen, so daß k_1 und v_1 für die Resultate des ersten Vergleichs und k_2 und v_2 für die des zweiten Vergleichs stehen. Analog dazu werden die durch den menschlichen Beobachter im Schritt 1 erkannten Verhalten mit V_{M_1} und die im Schritt 3 durch den Menschen erkannten Verhalten mit V_{M_2} bezeichnet.

Damit die Resultate mit den Ergebnissen der Verhaltens-Erkennung von Müller [43] verglichen werden können, werden die selben Spiele und Zeitintervalle als Untersuchungsintervalle verwendet. Es handelt sich dabei um 10 Spiele, von denen jeweils die Zeitintervalle (1400–1599), (3400–3599) und (5400–5599) untersucht werden (Spiele und Simulationsintervalle wurden in [43] „pseudozufällig“ ausgewählt).

Ergebnisse: In Tabelle 7.3 sind die durch den menschlichen Beobachter erkannten Verhalten (1. Evaluierungsschritt) und die durch den Coach erkannten Verhalten (2. Evaluierungsschritt) gegenübergestellt. Dabei handelt es sich um das Zeitintervall (1400–1599) eines der 10 untersuchten Fußballspiele (Arvand gegen Brainstormers01).

Beobachtung durch den Menschen						Analyse durch den Coach						
Name	von	bis	p_1	p_2	A.	Name	von	bis	p_1	p_2	A.	
Kampf	1400	1406	—	op_5	–	Kampf	1400	1406	—	op_5	–	√
Paß	1406	1413	op_5	—	F	Paß	1406	1413	op_5	—	F	√
Kampf	1413	1450	—	—	–	Kampf	1413	1450	—	—	–	√
Dribbeln	1473	1483	op_9	—	E	Dribbeln	1473	1483	op_9	—	E	√
Klären	1483	1528	op_9	mp_2	F	Klären	**1482**	**1527**	op_9	mp_2	F	√
Paß	1528	1540	mp_2	op_5	F	Paß	**1527**	1540	mp_2	op_5	F	√
Dribbeln	1540	1545	op_5	—	E	Dribbeln	1540	1545	op_5	—	E	√
Paß	1544	1553	op_5	op_9	E							–
						Paß	**1544**	**1558**	op_5	mp_2	F	+
Klären	1553	1559	op_9	mp_2	F							–
Paß	1559	1573	mp_2	mp_4	E	Paß	**1558**	**1572**	mp_2	mp_4	E	√
Paß	1573	1588	mp_4	mp_7	E	Paß	1573	**1587**	mp_4	mp_7	E	√
						Dribbeln	**1587**	**1590**	mp_7	—	E	+
Paß	1589	1600	mp_7	mp_{11}	E	Paß	1589	1600	mp_7	mp_{11}	E	√

Tabelle 7.3: Beispiel des Vergleichs der Beobachtungen von Mensch und Coach

Für jedes erkannte Verhalten sind jeweils dessen Name (Kampf steht für „Kampf um alleinige Ballkontrolle“), dessen Start- (von) und Endzeitpunkt (bis), dessen beteiligte Spieler (p_1, p_2) zum Start- bzw. Endzeitpunkt des Verhaltens insofern diese eindeutig sind sowie dessen Ausprägung (A.) hinsichtlich des Erfolges (‘E’ steht für erfolgreiches und ‘F’ für fehlerhaftes bzw. erfolgloses

Verhalten) angegeben. Bei dem Spiel wurde die Verhaltens-Modellierung aus Sicht des Teams Arvand durchgeführt, so daß mp_x Spieler des Teams Arvand und op_y Spieler des Teams Brainstormers01 bezeichnen.

Zur besseren Kenntlichmachung der Unterschiede in der Erkennung von Mensch und Coach, sind die durch den Coach als abweichend erkannte Eigenschaften der Verhalten fett dargestellt. In der rechten Spalte der Tabelle ist angegeben, ob

- die Beobachtungsergebnisse von Mensch und Coach übereinstimmen ($\surd$);
- der Coach ein Verhalten erkannt hat, welches nicht vom Menschen beobachtet wurde (+) bzw.
- der Mensch ein Verhalten erkannt hat, welches nicht vom Coach identifiziert wurde (–).

Für das in Tabelle 7.3 betrachtete Zeitintervall ergeben sich für die Kardinalitäten der durch Mensch, Coach bzw. durch beide erkannten Verhalten die Werte: $|V_{M_1}| = 12$, $|V_C| = 12$ und $|V_C \cap V_{M_1}| = 10$. Damit ergeben sich für die Korrektheit und Vollständigkeit die Werte: $k_1 = \frac{5}{6}$ und $v_1 = \frac{5}{6}$.

Bei einer erneuten Überprüfung für die abweichend erkannten Verhalten (3. Schritt) konnte die Interpretation durch den Coach in allen Fällen bestätigt werden. Damit ergeben sich für die Korrektheit und Vollständigkeit die Werte: $k_2 = 1$ und $v_2 = 1$.

In Tabelle 7.4 sind die Ergebnisse der manuellen Evaluierung der Verhaltens-Erkennung zusammengefasst. Für jedes der 10 untersuchten Spiele wurden für drei fixierte Zeitintervalle mit einer Gesamtzeit von 600 Takten pro Spiel die Werte für Korrektheit und Vollständigkeit bestimmt. In der Tabelle sind jeweils die zwei gegeneinander spielenden Mannschaften und die Werte k_1, v_1, k_2, und v_2 angegeben. In der letzten Zeile sind diese Werte für alle 6000 untersuchten Takte angegeben.

Nr.	Team A	Team B	k_1	v_1	k_2	v_2
1	WROCLAW	RoboLog2k1	0,867	0,867	0,978	0,978
2	RoboLog2k1	DrWeb	0,875	0,857	0,958	0,958
3	FCPortugal	UvA_Trilearn2001	0,9	0,88	1	1
4	LuckyLuebeck	Aras	0,929	0,907	0,976	0,953
5	FCPortugal	DrWeb	0,956	0,956	1	0,978
6	Aras	Arvand	0,982	0,965	1	1
7	DrWeb	UvA_Trilearn2001	0,98	0,962	1	1
8	Arvand	Brainstormers01	0,882	0,865	0,98	0,98
9	UvA_Trilearn2001	Arvand	0,943	1	0,981	1
10	DrWeb	VirtualWerder	0,886	0,861	0,971	0,971
	Spiele 1 bis 10		0,922	0,914	0,985	0,983

Tabelle 7.4: Korrektheit und Vollständigkeit der Verhaltens-Erkennung

Beim Vergleich der durch Mensch und Coach erkannten Verhaltens-Ausprägungen, über die Unterteilung in erfolglos und erfolgreich hinaus (es wurden insbesondere die einzelnen Paß- und Dribbel-Ausprägungen betrachtet), konnte eine überwiegend übereinstimmende Interpretation festgestellt werden. Während es beim ersten Vergleich teilweise noch Differenzen bei den Paß-Ausprägungen

Direktpaß und Indirekter Paß sowie bei den Dribbel-Ausprägungen „ständige Ballkontrolle“ und „vorübergehende Aufgabe der Ballkontrolle“ gab, konnte beim zweiten Vergleich die Interpretation des Coaches bestätigt werden. Damit verringern sich bei differenzierter Betrachtung dieser Verhaltens-Ausprägungen die Werte von v_1 und k_1 noch geringfügig, die Werte von v_2 und k_2 bleiben aber gleich.

Auswertung: Die Ergebnisse in Tabelle 7.4 zeigen, daß die Verhaltens-Erkennung durch den Coach überwiegend korrekt und vollständig ist. Mit durchschnittlichen Werten von über 0,98 für Korrektheit und Vollständigkeit beim 2. Vergleich bzw. von über 0,91 beim 1. Vergleich ist die Verhaltens-Erkennung des Coaches in Bezug auf die Qualität durchaus der Erkennung durch den Menschen ebenbürtig.

Die Unterschiede der Ergebnisse des ersten und zweiten Vergleichs können dabei sowohl als Fehler des Menschen bei der Verhaltens-Erkennung, als auch als unterschiedliche Interpretationsmöglichkeiten einzelner Handlungsweisen angesehen werden. Ausgehend von der vorliegenden Verhaltens-Spezifikation sind diese Unterschiede als Erkennungsfehler des Menschen zu betrachten.

Die menschlichen Fehler können dabei auf Unklarheiten bei der Ballkontrolle, auf Unklarheiten beim Ziel von Balltransfers sowie auf Unaufmerksamkeit zurückgeführt werden. Die Unklarheiten bei der Ballkontrolle resultieren

- in falsch erkannten Start- bzw. Endzeiten einzelner Verhalten,
- in der Erkennung mehrerer Verhalten, wobei nur ein Verhalten stattgefunden hat (z.B. Unterschied der Verhaltens-Erkennung in Tabelle 7.3) sowie
- in der Erkennung nur eines Verhaltens, wobei mindestens zwei Verhalten stattgefunden haben.

Unklarheiten beim Ziel von Balltransfers resultieren in Verwechslungen von Paß- bzw. Torschuß-Verhalten mit dem Klären-Verhalten. Aufgrund von Unaufmerksamkeit können einzelne kurze Dribbel-Verhalten übersehen werden (z.B. letzter Unterschied der Verhaltens-Erkennung in Tabelle 7.3).

Die Differenzen der Korrektheit und Vollständigkeit zwischen dem zweiten und ersten Vergleich können damit als ein Fehlerwert der menschlichen Beobachtung und die Differenzen zwischen 1 und den Werten des zweiten Vergleichs als ein Fehlerwert der Beobachtung durch den Coach angesehen werden. Damit ist der Fehlerwert des Coaches wesentlich geringer als der Fehlerwert des Menschen.

Die wenigen Fehler bei der Verhaltens-Erkennung durch den Coach lassen sich auf die zu genaue Auslegung der Ballkontrolle durch Spieler zurückführen. So erkennt der Coach das Kicken eines Spielers nur, wenn sich im Anschluß die Geschwindigkeit des Balles, über eine zufällige Bewegungsänderung hinausgehend, geändert hat. Damit wird in einigen Fällen der Start- bzw. der Endzeitpunkt eines Verhaltens nicht korrekt erkannt. Desweiteren kann es vorkommen, daß damit zwei Verhalten nur als eines erkannt werden. Auch das Verhalten „Kampf um alleinige Ballkontrolle“ wird öfter erkannt, als es tatsächlich stattfindet.

Basierend auf den Erfahrungen aus der Diplomarbeit von Müller [43], konnten Korrektheit und Vollständigkeit der Verhaltens-Erkennung auch bei Einbeziehung spezialisierterer Verhaltens-Muster weiter verbessert werden. Müller spezifiziert die Verhalten auf andere Art und Weise. Er führt die Verhaltens-Muster Paß, Dribbeln, Torschuß und Abfangen ohne Ausprägungen ein. Das Verhaltens-Muster Abfangen steht dabei für fehlerhafte Verhalten, bei denen ein gegnerischer Spieler die Kontrolle über den Ball erlangt. Auch die anderen drei Verhaltens-Muster sind nicht mit den in dieser Arbeit spezifizierten Verhaltens-Mustern identisch.

Da die gleichen Spiele wie in der Diplomarbeit von Müller untersucht wurden, können die Werte für Korrektheit und Vollständigkeit dennoch verglichen werden. In Tabelle 7.5 sind die Ergebnisse der Verhaltens-Erkennung dieser Arbeit den Ergebnissen der Arbeit von Müller gegenübergestellt. Neben den Werten für Korrektheit und Vollständigkeit des ersten und zweiten Vergleichs sind noch die in den 10 Spielen insgesamt vom Coach erkannten Verhalten aufgeführt.

	Ansatz Müller [43]	Ansatz Wendler
durch Coach erkannte Verhalten	332	475
Korrektheit 1. Vergleich (k_1)	0,904	0,922
Vollständigkeit 1. Vergleich (v_1)	0,85	0,914
Korrektheit 2. Vergleich (k_2)	0,976	0,985
Vollständigkeit 2. Vergleich (v_2)	0,928	0,983

Tabelle 7.5: Vergleich der Ergebnisse der Verhaltens-Erkennung

Durch die Verwendung speziellerer Verhaltens-Muster können nicht nur die erkannten Verhalten genauer beschrieben werden, sondern wie aus Tabelle 7.5 zu entnehmen ist, konnte auch die Korrektheit und vor allem die Vollständigkeit der Verhaltens-Erkennung verbessert werden.

Mit der hohen Genauigkeit der Verhaltens-Erkennung steht einer Verwendung der erkannten Verhalten zur Generierung von Verhaltens-Modellen nichts im Wege. Über diese Verwendung hinaus können die Ergebnisse der Verhaltens-Bestimmung auch für Kommentatoren, zur Generierung von Statistiken und zur automatischen Identifikation von fehlerbehafteten bzw. erfolglosen Verhalten verwendet werden.

7.3 Evaluierung der Verhaltens-Vorhersage

Zur Evaluierung der Verhaltens-Vorhersage wird das vom Coach vorhergesagte Verhalten mit dem daran anschließend beobachteten Verhalten verglichen. Da die Verhaltens-Erkennung des Coaches sehr zuverlässig ist, wird das durch den Coach erkannte Verhalten zum Vergleich verwendet. Damit kann die Evaluierung der Verhaltens-Vorhersage automatisch durchgeführt werden.

Die Evaluierung der Verhaltens-Vorhersage wird anhand des Verhaltens-Musters Paß (d.h. $\mathbb{NB} = \{\text{pass}\}$) vorgenommen. Da ein Vergleich der Vorhersage mit einem ausgeführten Paß-Verhalten erst nach der Erkennung des Verhaltens möglich ist, wird die Verhaltens-Vorhersage im Anschluß an die Erkennung eines Paß-Verhaltens durchgeführt. Dabei werden Vorhersagen nur

für die Entscheidungs-Situationen vorgenommen, die tatsächlich zu einem Paß-Verhalten durch die IP geführt haben. Diese Einschränkung ist nötig, da nur die Vorhersage für Paß-Verhalten implementiert wurde und damit keine Fälle der anderen Verhaltens-Muster als Handlungsalternativen zur Verfügung stehen.

Für den Ähnlichkeitsvergleich zwischen vorhergesagten und beobachteten Verhalten muß die dabei verwendete Gewichtesequenz (vgl. Formel 5.16) spezifiziert werden. Für die Evaluierung der Verhaltens-Vorhersage wird als Gewichtesequenz für die Paß-Verhalten die Sequenz $\mathtt{W}_{\mathtt{B}}^{\text{pass}} = (3, 5, 0, 0, 0)$ verwendet, d.h. es wird großen Wert auf den Winkel des Ballgeschwindigkeitsvektors (Gewicht 5) und die Länge des Ballgeschwindigkeitsvektors (Gewicht 3) gelegt. Den anderen Eigenschaften des Paß-Verhaltens (vgl. Abschnitt 5.3.1 auf Seite 112) wird im Rahmen der Evaluierung keine Bedeutung beigemessen.

Zuerst werden mit der **Genauigkeit** und der **Wirksamkeit** die zur Evaluierung der Verhaltens-Vohersage verwendeten Begriffe eingeführt (Abschnitt 7.3.1). Danach wird die Bestimmung der Gewichte für die Auslöser evaluiert. Dabei wird u.a. ermittelt, welchen Einfluß die Verwendung von angepaßten Gewichten für die Verhaltens-Vorhersage hat (Abschnitt 7.3.2). Es folgt eine Untersuchung der Abhängigkeit der Vorhersage-Genauigkeit und der Vorhersage-Wirksamkeit von der Anzahl der verwendeten Auslöser-Verhaltens-Fälle (Abschnitt 7.3.3). Dann wird für eine größere Menge von Teams die Verhaltens-Vorhersage evaluiert sowie die Vorhersage-Genauigkeit des Paß-Empfängers untersucht (Abschnitt 7.3.4). Schließlich wird ein Überblick über weitere durchgeführte Analysen der Verhaltens-Vorhersage gegeben (Abschnitt 7.3.5).

7.3.1 Genauigkeit und Wirksamkeit

Zur Evaluierung muß das vorhergesagte mit dem beobachteten Verhalten verglichen werden. Dazu bietet sich das für Verhalten definierte Ähnlichkeitsmaß an. Basierend auf der Ähnlichkeit zwischen dem vorhergesagten Verhalten BE^P und dem erkannten Verhalten BE^R wird die **Genauigkeit** dieser Verhaltens-Vorhersage bestimmt (siehe Formel 7.5). Es wird also nicht ermittelt, ob die Vorhersage korrekt oder nicht korrekt ist, sondern wie ähnlich sie dem tatsächlich durchgeführten Verhalten ist.

$$\mathit{Genauigkeit}(BE^P, BE^R) = \text{similarity}_{\mathrm{B}}\left(BE^P, BE^R\right) \tag{7.5}$$

Damit ist die Genauigkeit der Verhaltens-Vohersage vom verwendeten Ähnlichkeitsmaß abhängig. Es könnte ein Ähnlichkeitsmaß definiert werden, bei dem die Ähnlichkeit zweier beliebiger Verhalten immer den Wert 1 ergibt. Die Genauigkeit der Verhaltens-Vorhersage wäre dann immer maximal. Der vorgestellte Ansatz hätte aber auch keinen Vorteil gegenüber einer zufälligen Verhaltensauswahl.

Mit der **Wirksamkeit** soll ein Wert definiert werden, der vom verwendeten Ähnlichkeitsmaß weitgehend unabhängig ist. Insbesondere soll mit der Wirksamkeit ausgesagt werden, wie groß der Vorteil des verwendeten Ansatzes gegenüber einer zufälligen Auswahl eines Verhaltens ist. Dazu wird der Wert $avgSim_{\mathtt{behaviors}_E}$ eingeführt, der die durchschnittliche Ähnlichkeit aller Verhaltens-Essenzen $BE \in \mathtt{behaviors}_E$ bzgl. der beobachteten Verhaltens-Essenz

BE^R angibt (siehe Formel 7.6 für endliche Mengen $\texttt{behaviors}_E$). Bei dem für die Evaluierung verwendeten Ähnlichkeitsmaß kann $avgSim_{\texttt{behaviors}_E}$ Werte zwischen 0,1297 und 0,1725 annehmen[8].

$$avgSim_{BE}(BE^R) = \frac{1}{|\texttt{behaviors}_E|} \sum_{BE \in \texttt{behaviors}_E} \text{similarity}_\text{B}\,(BE, BE^R) \quad (7.6)$$

Die Wirksamkeit einer Verhaltens-Vorhersage wird als das Verhältnis zwischen der Vorhersage-Genauigkeit und der durchschnittlichen Ähnlichkeit aller Verhaltens-Essenzen bzgl. der beobachteten Verhaltens-Essenz definiert (siehe Formel 7.7).

$$Wirksamkeit_{BE}(BE^P, BE^R) = \frac{Genauigkeit(BE^P, BE^R)}{avgSim_{BE}(BE^R)} \quad (7.7)$$

Damit kann die Vorhersage-Wirksamkeit zwischen 0 (*Genauigkeit* ist 0) und 7,7101 (*Genauigkeit* ist 1 und $avgSim_{BE}$ ist 0,1297) liegen. Ein Wert kleiner als 1 stellt dabei eine ungünstige Vorhersage dar. Werte größer 1 kennzeichnen einen Vorteil der Verhaltens-Vorhersage gegenüber einer zufälligen Verhaltensauswahl.

Die Begriffe Genauigkeit und Wirksamkeit werden im weiteren für Durchschnittswerte mehrerer Vorhersagen (meist für ein Spiel) verwendet. Dazu wird jeweils das arithmetische Mittel der Genauigkeiten und der Wirksamkeiten aller durchgeführten Vorhersagen gebildet. Einzelne Vorhersagen können dabei durchaus eine Wirksamkeit von kleiner Eins haben, sehr gute Vorhersagen sollten solche fehlerhaften Vorhersagen wieder ausgleichen.

Neben der durchschnittlichen Ähnlichkeit des beobachteten Verhaltens bezogen auf alle Verhaltens-Essenzen $avgSim_{BE}$, kann auch die durchschnittliche Ähnlichkeit des beobachteten Verhaltens bezogen auf die Verhaltens-Essenzen aller Fälle der Fallbasis $avgSim_{\texttt{cases}}$ als Vergleichswert verwendet werden (siehe Formel 7.8). Die Werte von $avgSim_{\texttt{cases}}$ sind abhängig von der verwendeten Fallbasis und können prinzipiell zwischen 0 und 1 liegen.

$$avgSim_{\texttt{cases}}(BE^R) = \frac{1}{|caseBase|} \sum_{(T,BE) \in caseBase} \text{similarity}_\text{B}\,(BE, BE^R) \quad (7.8)$$

Die **Wirksamkeit bezogen auf die Fallbasis**, wird als Verhältnis zwischen der Genauigkeit und $avgSim_{\texttt{cases}}$ definiert (siehe Formel 7.9) und kann damit beliebige Werte größer gleich Null annehmen.

$$Wirksamkeit_{\texttt{cases}}(BE^P, BE^R) = \frac{Genauigkeit(BE^P, BE^R)}{avgSim_{\texttt{cases}}(BE^R)} \quad (7.9)$$

Auch die Wirksamkeit bezogen auf die Fallbasis, wird nachfolgend als Durchschnittswert mehrerer Vorhersagen verwendet. Die allgemeine Wirksamkeit $Wirksamkeit_{BE}$ stellt den Vorteil des kompletten Ansatzes dieser Arbeit gegenüber einer zufälligen Auswahl des Verhaltens dar. Die Wirksamkeit bezogen

[8] Diese Werte wurden auf Basis des verwendeten Ähnlichkeitsmaßes berechnet.

auf die Fallbasis $Wirksamkeit_{\texttt{cases}}$, kennzeichnet den Vorteil der Bestimmung von Auslösern und deren Verwendung zur Vorhersage gegenüber einer zufälligen Auswahl von in der Vergangenheit beobachteten Verhalten der IP.

Für die Evaluierungen in den folgenden drei Abschnitten wurden Verhaltens-Modelle mehrerer Teams generiert. Der Nutzen eines jeden Verhaltens-Modells für die Vorhersage des Verhaltens von Spielern des modellierten Teams wird anhand der durchschnittlichen Genauigkeits- und/oder Wirksamkeitswerte evaluiert. Die zur Generierung der Modelle verwendeten Daten sind in den jeweiligen Abschnitten aufgeführt.

7.3.2 Bestimmung der Gewichte für Auslöser

Versuchsplanung: In diesem Abschnitt wird die Bestimmung angepaßter Auslöser-Gewichte mittels *Cross Validation* (siehe Abschnitt 5.5.4.2 auf Seite 122) auf ihren Nutzen hin untersucht. Dazu werden automatisch generierte Gewichtesequenzen mit manuell festgelegten Gewichtesequenzen verglichen. Desweiteren wird die Abhängigkeit der Vorhersage-Genauigkeit von der Anzahl der verwendeten Gewichtseinheiten (vgl. Abschnitt 5.5.4.3 auf Seite 124) untersucht.

Als Fallbasis werden die aus den 5 Vorrunden-Spielen generierten Fälle eines Teams verwendet. Für die Bestimmung der Gewichtesequenz werden die Situations-Verhaltens-Paare eines der 5 Vorrunden-Spiele genutzt. Auf Basis dieser Daten werden für verschiedene Anzahlen von Gewichtseinheiten jeweils die Gewichtesequenz für Auslöser bestimmt. Es werden die Anzahlen $5, 8, 10, 12, 15, 20, 25$ und 30 betrachtet.

Die so ermittelten Gewichtesequenzen werden evaluiert, indem für jedes der 5 Vorrunden-Spiele die Vorhersage-Genauigkeit bestimmt wird[9]. Zum Vergleich werden auf die gleiche Art und Weise zwei manuell generierte Gewichtesequenzen evaluiert. Bei den manuell ausgewählten Gewichtesequenzen $\mathtt{W}_{\mathtt{T}}^{name}$ handelt es sich um die Sequenzen M1 $= (1, 1, 1, 1, 1, 1)$ und M2 $= (3, 5, 2, 1, 2, 1)$[10]. Bei der ersten Sequenz haben alle primären Attribute des Auslöser-Musters gleiche Bedeutung, mit der zweiten Sequenz werden einzelne Attribute stärker gewichtet[11] (vgl. Abschnitt 5.1.2 auf Seite 103). Das Paß-Auslöser-Muster besteht aus vier verschiedenen primären Attributen (die letzten beiden Attribute sind mit den Attributen davor identisch), so daß die erste Sequenz aus 4 und die zweite Sequenz aus 11 Gewichtseinheiten gebildet wurde.

Die Evaluierung des Einflusses der automatischen Bestimmung der Gewichtesequenzen wird für die zwei Teams FCPortugal und Brainstormers01 vorgenommen. Die beste automatisch bestimmte Gewichtesequenz (größte Vorhersage-Genauigkeit aller bestimmten Gewichtesequenzen) eines Teams wird jeweils auch beim jeweiligen anderen Team untersucht.

[9] Wie bei der Bestimmung der Auslöser-Gewichte werden auch bei den Evaluierungen jeweils die Fälle deren Auslöser identisch mit der Entscheidungs-Situation sind, nicht für die Verhaltens-Vorhersage verwendet.

[10] M2 wurde aufgrund von Überlegungen und ersten Experimenten ausgewählt.

[11] Die Kickerposition erhält 3, die relative Position des Empfängers 5, die Winkel zum Gegner jeweils 2 und die Entfernungen zum Gegner jeweils eine Gewichtseinheit.

Zusammengefasst hat das Experiment folgende Eigenschaften:
- es wird bzgl. der Gewichtesequenzen variiert
- die Fallbasis wird aus den 5 Vorrundenspielen pro Team generiert
- die Gewichtesequenzen werden für ein zufälliges Vorrundenspiel des Teams bestimmt
- die Vorhersage wird mit allen 5 Vorrundenspielen des Teams evaluiert
- die Prozedur wird für die Teams FCPortugal und Brainstormers01 durchgeführt

Ergebnisse: In Tabelle 7.6 sind die so ermittelten Gewichtesequenzen je Team (inklusive der manuell ausgewählten) zu sehen. Desweiteren ist die jeweils längste Dauer der Bestimmung der automatisch generierten Gewichtesequenzen angegeben. Aus der Tabelle ist bereits ersichtlich, daß die Gewichtesequenzen vom modellierten Team abhängen. So hat für das Team FCPortugal anscheinend der Vektor zum Empfänger eine sehr große Bedeutung, wohingegen das Paß-Verhalten des Teams Brainstormers01 u.a. auch stark von der Position des Kickers abhängt. Die jeweils Beste der automatisch evaluierten Gewichtesequenzen ist fett dargestellt und wurde zusätzlich beim jeweilig anderen Team aufgenommen und mit 'B' gekennzeichnet.

Anzahl der Gewichtseinheiten	Gewichtesequenz FCPortugal (689 Fälle)	Gewichtesequenz Brainstormers01 (686 Fälle)	max. Zeit in Sekunden
M1 (manuell)	$(1,1,1,1,1,1)$		—
M2 (manuell)	$(3,5,2,1,2,1)$		—
5	$(2,3,0,0,0,0)$	$(1,1,3,0,3,0)$	396
8	$(3,4,1,0,1,0)$	$(1,1,6,0,6,0)$	1.171
10	$(\mathbf{3},\mathbf{6},\mathbf{1},\mathbf{0},\mathbf{1},\mathbf{0})$	$(3,3,3,1,3,1)$	2.031
12	$(5,5,2,0,2,0)$	$(3,4,4,1,4,1)$	3.249
15	$(5,7,3,0,3,0)$	$(10,3,2,0,2,0)$	5.824
20	$(4,13,3,0,3,0)$	$(5,7,7,1,7,1)$	12.579
25	$(5,16,4,0,4,0)$	$(6,10,8,1,8,1)$	23.227
30	$(9,17,4,0,4,0)$	$(\mathbf{10},\mathbf{10},\mathbf{9},\mathbf{1},\mathbf{9},\mathbf{1})$	38.669
B	$(10,10,9,1,9,1)$	$(3,6,1,0,1,0)$	—

Tabelle 7.6: Ermittelte und zur Evaluierung verwendete Gewichtesequenzen

Die Vorhersage-Genauigkeiten der fünf zur Evaluierung verwendeten Spiele für die in Tabelle 7.6 aufgeführten Gewichtesequenzen sind in den Abbildungen 7.2 (Team Brainstormers01) und 7.3 (Team FCPortugal) zu sehen.

Für jede Gewichtesequenz wurde für jedes evaluierte Spiel eine Genauigkeitsmessung vorgenommen. Diese Genauigkeitswerte sind in der Abbildung durch helle, kleine Kreise dargestellt. Dabei wurden die Genauigkeitswerte des zur Generierung der Auslöser-Gewichtesequenzen verwendeten Spieles mit einer hellen Strichellinie verbunden. Die durchschnittlichen Genauigkeiten für jede Gewichtesequenz sind mit dunklen Rauten eingezeichnet, die mit einer Linie verbunden wurden.

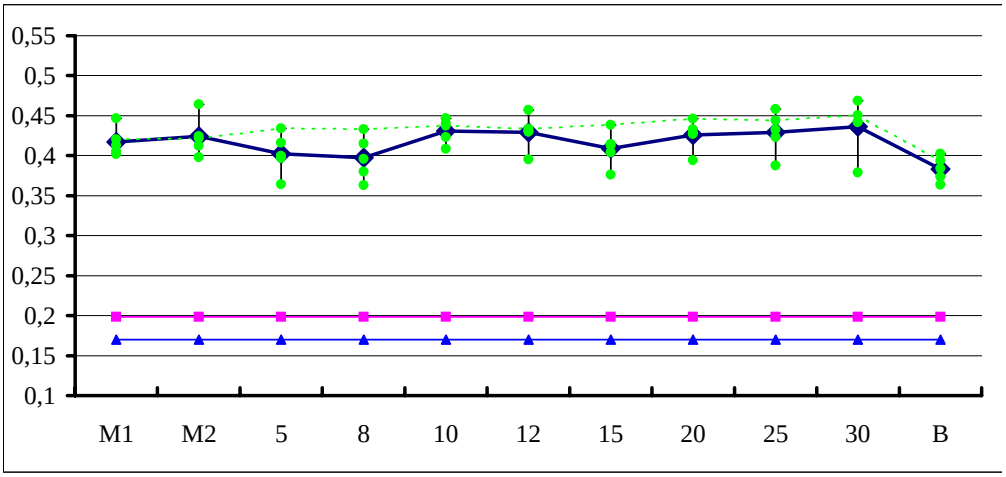


Abbildung 7.2: Genauigkeit der Verhaltens-Vorhersage in Abhängigkeit von verschiedenen Gewichtesequenzen für das Team Brainstormers01

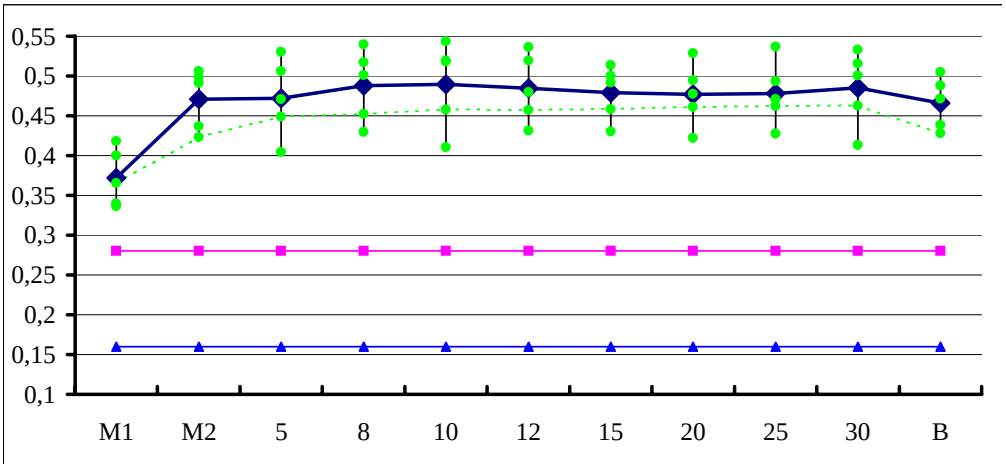


Abbildung 7.3: Genauigkeit der Verhaltens-Vorhersage in Abhängigkeit von verschiedenen Gewichtesequenzen für das Team FCPortugal

Die über alle Spiele gemittelten durchschnittlichen Ähnlichkeiten $avgSim_{BE}$ und $avgSim_{\mathsf{cases}}$ sind durch Dreiecke bzw. Quadrate eingezeichnet. Diese Werte sind für alle Gewichte dieselben, da sie nicht von den verwendeten Auslöser-Gewichtesequenzen abhängen.

Auswertung: In den Abbildungen 7.2 und 7.3 ist zu sehen, daß eine Erhöhung der Gewichtseinheiten nicht wesentlich zur Vorhersage-Genauigkeit beiträgt. Das gleiche gilt für die Vorhersage-Wirksamkeit (infolge der Unabhängigkeit der durchschnittlichen Ähnlichkeiten bzgl. der Gewichtesequenzen). Bei Verwendung von 12 Gewichtseinheiten konnten, auch über die in den Abbildungen dargestellten Experimente hinaus, entweder die höchsten oder nur geringfügig schlechtere Genauigkeitswerte ermittelt werden. Für die Experimente in den folgenden Abschnitten werden daher zur Bestimmung der Gewichtesequenzen immer 12 Gewichtseinheiten verwendet.

Beim Team Brainstormers01 erreicht man bereits mit den beiden manuellen Gewichtesequenzen M1 und M2 Genauigkeitsresultate, die nur noch unwesentlich schlechter sind als die automatisch generierten Sequenzen. Bei der Verwendung der optimalen Gewichtesequenz B des Teams FCPortugal ergeben sich aber schlechtere Genauigkeitswerte.

Für das Team FCPortugal kann dagegen die beste Gewichtesequenz B des Teams Brainstormers01 verwendet werden, ohne daß sich die Genauigkeitswerte stark verschlechtern. Die manuelle Gewichtesequenz M1 führt jedoch zu wesentlich schlechteren Vorhersage-Ergebnissen als die automatisch generierten Gewichtesequenzen. Obwohl die Gewichtesequenz M2 zur Verhaltens-Vorhersage des Teams FCPortugal geeignet ist, kann prinzipiell auch diese Gewichtesequenz für andere Teams ungeeignet sein. Mit der automatischen Bestimmung von Gewichtesequenzen kann die Verwendung ungeeigneter Gewichtesequenzen weitgehend ausgeschlossen werden.

Die Bedeutung der automatischen Ermittlung der Bonus-Werte wurde auf ähnliche Art und Weise untersucht. Der Einfluß der Bonus-Werte auf die Vorhersage-Genauigkeit war bei den durchgeführten Experimenten sehr gering.

7.3.3 Abhängigkeit von der Anzahl der Fälle

Versuchsplanung: Die Untersuchung der Abhängigkeit der Verhaltens-Vorhersage von der Anzahl der zur Verfügung stehenden Auslöser-Verhaltens-Fälle ist Gegenstand dieses Abschnittes. Dazu werden verschieden große Fallbasen generiert und auf deren Grundlage die Vorhersage-Genauigkeiten zweier Teams bestimmt.

Es ist zu erwarten, daß die Vorhersage genauer wird, wenn mehr Fälle für die Vorhersage zur Verfügung stehen. Mit einer größeren Fallbasis steigt die durchschnittliche Bewertung σ_i des zur Vorhersage verwendeten bewerteten Falles RC_i (vgl. Formel 6.2 auf Seite 133). Unter der Vorraussetzung, daß ein Team in ähnlichen Entscheidungs-Situationen auch ähnliches Verhalten ausführt, sollte sich daher die Verhaltens-Vorhersage verbessern, wenn die Fallbasis vergrößert wird.

Die Untersuchung der Abhängigkeit der Vorhersage von der Größe der verwendeten Fallbasis wird für die Teams Brainstormers01 und Arvand durchgeführt. Beide Teams bestritten während des Turniers mit 10 Spielen die meisten Spiele. Jeweils die ersten 8 dieser Spiele wurden zur Generierung der 8 Fallbasen verwendet, wobei die j-te Fallbasis aus den ersten j der 8 Spiele generiert wurde. Die Gewichtesequenz und die Bonus-Werte wurden jeweils für ein zufälliges der 8 Spiele bestimmt. Zur Evaluierung der Verhaltens-Vorhersage stehen die beiden letzten Spiele je Team zur Verfügung.

Zusammengefasst hat das Experiment folgende Eigenschaften:

- es wird bzgl. der Größe der verwendeten Fallbasen variiert
- die Fallbasen werden aus 1 bis 8 Spielen generiert
- die Gewichtesequenzen werden für ein zufälliges der 8 Spiele bestimmt
- die Vorhersage wird mit den beiden letzten Spielen des Teams evaluiert
- die Prozedur wird für die Teams Brainstormers01 und Arvand durchgeführt

Ergebnisse: In den Abbildungen 7.4 (Team Brainstormers01) und 7.5 (Team Arvand) sind die Vorhersage-Genauigkeiten der 2 letzten Spiele des jeweiligen Teams in Abhängigkeit von der Größe der verwendeten Fallbasis dargestellt. Die Größe der Fallbasis ist dabei durch die Anzahl ihrer Auslöser-Verhaltens-Fälle (auf der waagerechten Achse für die 8 Fallbasen abgetragen) repräsentiert.

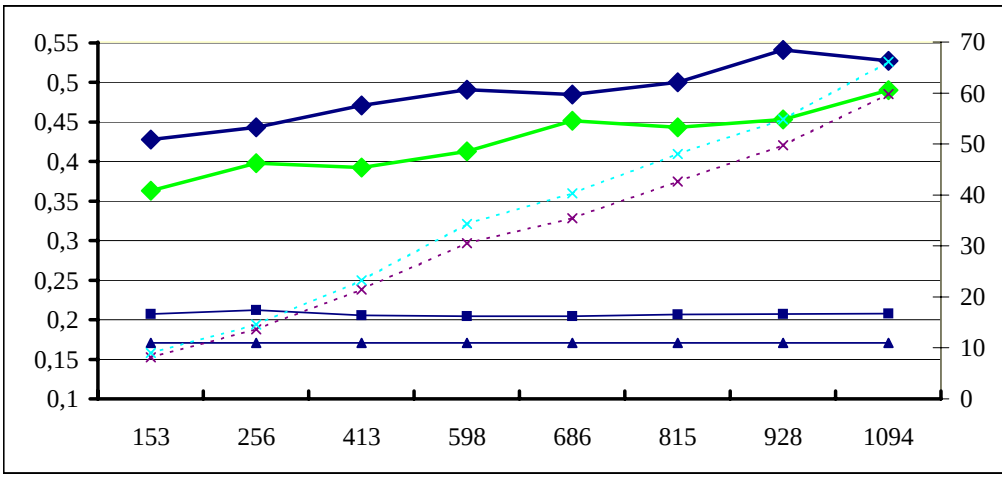


Abbildung 7.4: Vorhersage-Genauigkeit in Abhängigkeit von der Größe der verwendeten Fallbasis (Anzahl der Fälle) für das Team Brainstormers01

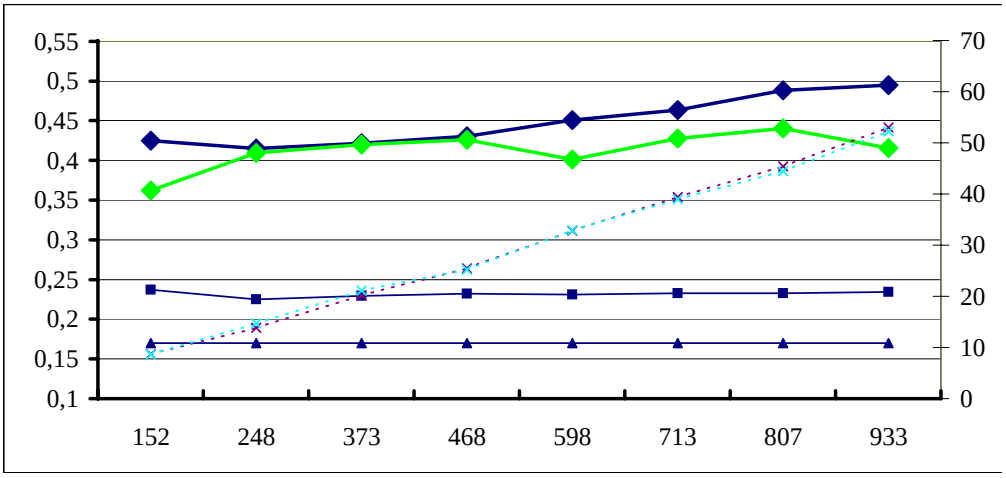


Abbildung 7.5: Vorhersage-Genauigkeit in Abhängigkeit von der Größe der verwendeten Fallbasis (Anzahl der Fälle) für das Team Arvand

Die Vorhersage-Genauigkeiten der beiden evaluierten Spiele sind durch Rauten dargestellt. Für eines der Spiele wurden dunkle (vorletztes Spiel des Teams im Turnier) und für das andere helle Rauten (letztes Spiel) verwendet. Zur Kennzeichnung der Tendenz wurden die Genauigkeitswerte jeweils durch eine Linie verbunden.

Die zur Vorhersage benötigten durchschnittlichen Zeiten je Spiel und Fallbasis sind durch kleine Kreuze dargestellt (dunkle Kreuze für das vorletzte und helle für das letzte Spiel), die durch eine Strichellinie verbunden sind. Die entsprechenden Werte für die Zeiten sind an der rechten Achse in Millisekunden abgetragen.

Für eines der Spiele sind außerdem die durchschnittlichen Ähnlichkeiten $avgSim_{BE}$ und $avgSim_{\texttt{cases}}$ durch Dreiecke bzw. Quadrate eingezeichnet (die durchschnittlichen Ähnlichkeiten des anderen Spieles weisen annähernd die gleichen Werte auf). Die durchschnittlichen Ähnlichkeiten $avgSim_{BE}$ sind für alle Vorhersagen gleich. Die durchschnittlichen Ähnlichkeiten $avgSim_{\texttt{cases}}$ unterliegen dagegen geringen Schwankungen.

Auswertung: In den beiden Abbildungen ist zu sehen, daß mit der Vergrößerung der Fallbasis durch Aufnahme zusätzlicher Fälle die Vorhersage-Genauigkeit in geringem Umfang gesteigert werden kann. Eine genauere Betrachtung der Abbildungen legt die Vermutung nahe, daß bei 900 bis 1000 Fällen eine gewisse Stagnation bei der Steigerung der Genauigkeit einsetzt. Bei weiteren Untersuchungen hat sich dieser Eindruck verstärkt.

Desweiteren kann festgestellt werden, daß die Aufnahme neuer Fälle nicht notwendigerweise zu einer Steigerung der Vorhersage-Genauigkeit führt. Die Steigerung ist davon abhängig, inwieweit die zusätzlich aufgenommenen Fälle zur Vorhersage der evaluierten Spiele geeignet sind. So sind in den Abbildungen trotz stetiger Vergrößerung der Fallbasis zwischenzeitlich auch geringfügig schlechtere Vorhersage-Genauigkeiten zu beobachten.

Für die Vorhersage-Wirksamkeiten $Wirksamkeit_{BE}$ und $Wirksamkeit_{\texttt{cases}}$ gilt das gleiche wie für die Vorhersage-Genauigkeit, da die durchschnittlichen Ähnlichkeiten für die evaluierten Spiele größtenteils unabhängig von der verwendeten Fallbasis sind.

7.3.4 Vorhersage des Paß-Verhaltens und des Paß-Empfängers

Versuchsplanung: In diesem Abschnitt wird für alle Teams, die die Finalrunde erreicht haben, die Vorhersage des Paß-Verhaltens und die Vorhersage des Paß-Empfängers in Abhängigkeit vom modellierten Team evaluiert.

Verhalten kann unterschiedlich detailliert beschrieben werden. Dafür wurden im Abschnitt 4.3.2 auf Seite 56 mehrere Stufen zur Beschreibung von Verhalten herausgearbeitet. Neben der Vorhersage aller charakteristischen Werte der Verhalten in Form von Verhaltens-Essenzen („konkrete Beschreibung“) wird in diesem Abschnitt auch die Vorhersage der Verhalten in Form des Paß-Empfängers („Quelle-Ziel-Beschreibung“) durchgeführt und evaluiert.

Für jedes der 8 Teams, die die Finalrunde erreicht haben, wird aus deren 5 Vorrundenspielen eine Fallbasis generiert. Auf Basis eines zufällig ausgewählten Spieles der Vorrunde werden jeweils die Gewichtesequenz und die Bonus-Werte ermittelt. Für die Evaluierung der Verhaltens-Vorhersage werden die Spiele der Endrunde verwendet.

Zusammengefasst hat das Experiment folgende Eigenschaften:
- es wird bzgl. der Teams variiert
- die Fallbasis wird aus den 5 Vorrundenspielen pro Team generiert
- die Gewichtesequenzen werden für ein zufälliges Vorrundenspiel des Teams bestimmt
- die Vorhersage wird mit allen Spielen der Finalrunde evaluiert
- die Prozedur wird für alle 8 Teams, die die Finalrunde erreicht haben, durchgeführt

Ergebnisse: In den folgenden Abbildungen sind Vorhersage-Resultate für verschiedene Teams dargestellt. Die Teams sind dabei durch ihre ersten drei Buchstaben gekennzeichnet. Eine Zuordnung dieser Kürzel zu den Teamnamen ist in Tabelle 7.7 gegeben. Die Größen der aus den 5 Vorrundenspielen generierten Fallbasen und die für die durchgeführten Vorhersagen durchschnittlich (avg) sowie maximal (max) benötigten Zeiten, sind ebenso aufgeführt. Desweiteren sind die Platzierungen der Teams bei den *German Open* angegeben.

Platz	Teamname	Kürzel	Anzahl Fälle in Fallbasis	Vorhersage-Zeit in Millisekunden	
				avg	max
1	FCPortugal	fcp	689	40,2	69
2	Brainstormers01	bra	686	37,9	71
3	Arvand	arv	598	32,0	67
4	DrWeb	drw	530	31,9	66
5	Aras	ara	518	29,4	60
5	UvA_Trilearn2001	uva	498	29,7	51
7	RoboLog2k1	rob	437	23,9	36
7	MRB	mrb	526	31,7	56

Tabelle 7.7: Größe der generierten Fallbasen sowie Vorhersage-Zeiten mehrerer Teams

In der Abbildung 7.6 auf der nächsten Seite sind die Genauigkeiten der Vorhersage des Paß-Empfängers für verschiedene Teams zu sehen. Die einzelnen Genauigkeitsmessungen pro Spiel sind durch helle Punkte und der Mittelwert über alle evaluierten Spiele pro Team ist jeweils durch eine dunkle Raute dargestellt.

Analog zu den durchschnittlichen Ähnlichkeiten bei Verhalten kann auch für die Vorhersage des Paß-Empfängers ausgesagt werden, wie erfolgsversprechend die zufällige Wahl des Paß-Empfängers ist. Da jeder Spieler 10 Mitspieler hat, ergibt sich bei der zufälligen Auswahl des Paß-Empfängers immer eine Erfolgswahrscheinlichkeit von 0,1. Dieser Wert ist für jedes Team mit einem Dreieck eingezeichnet.

Im Spiel kommen aber nicht alle Mitspieler als Paß-Empfänger in Frage, z.B. weil sie zu weit vom Kicker entfernt sind. So werden, abhängig von der Situation bei der Verhaltens-Vorhersage, nur Auslöser für potentielle Paß-Empfänger bestimmt. Auf Basis der potentiellen Paß-Empfänger kann auch

eine zufällige Auswahl erfolgen. Die durchschnittlichen Erfolgswahrscheinlichkeiten pro Team für eine solcherart zufällige Auswahl sind in der Abbildung mit hellen Quadraten eingezeichnet.

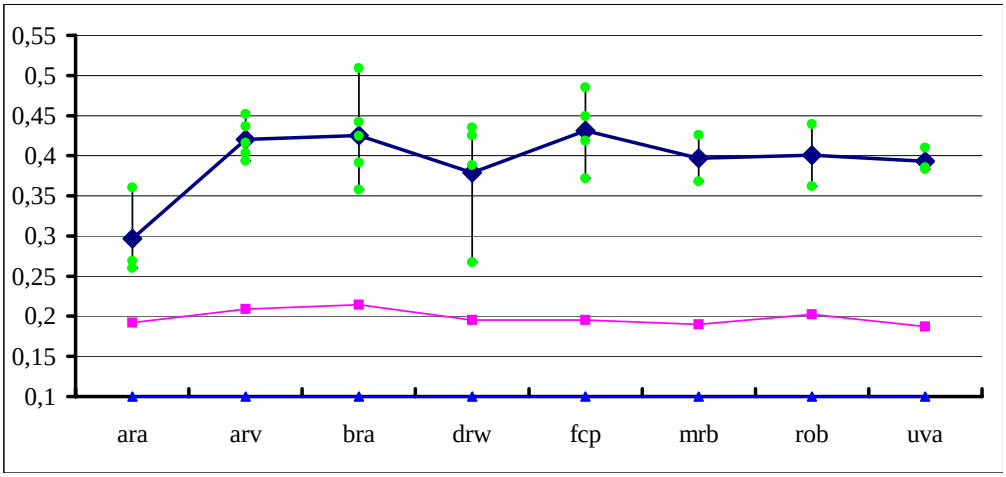


Abbildung 7.6: Genauigkeit der Vorhersage des Paß-Empfängers für verschiedene Teams

Die Genauigkeitsergebnisse der Verhaltens-Vorhersage sind in Abbildung 7.7 wiedergegeben. Auch hier kennzeichnet jeder helle Kreis ein Genauigkeitsergebnis eines Spiels. Deren Durchschnittswerte sind durch dunkle Rauten und die durchschnittlichen Ähnlichkeiten $avgSim_{BE}$ und $avgSim_{\texttt{cases}}$ durch Dreiecke bzw. Quadrate eingezeichnet.

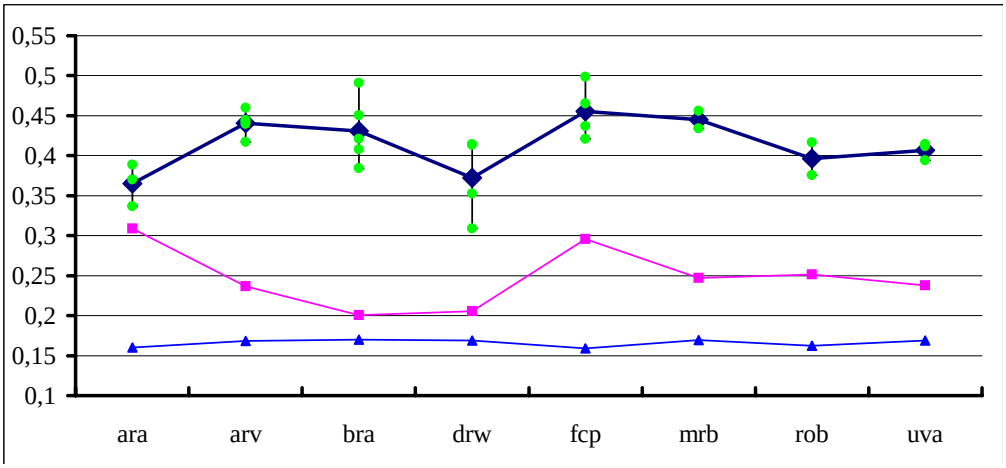


Abbildung 7.7: Genauigkeit der Verhaltens-Vorhersage für verschiedene Teams

Auswertung Vorhersage-Zeit: Die große Abweichung der maximalen Vorhersage-Zeit von der durchschnittlichen Vorhersage-Zeit (siehe Tabelle 7.7) ergibt sich aus dem Fakt, daß abhängig von der Situation unterschiedlich viele

Paß-Auslöser mit der Funktion identify_S generiert werden. Die Entscheidung, ob ein Spieler der eigenen Mannschaft als potentieller Paß-Empfänger betrachtet wird (ein Auslöser für ihn generiert wird), hängt u.a. von seiner Entfernung zum Kicker ab. Im ungünstigsten Fall muß für jeden Mitspieler ein Auslöser ermittelt werden. Für Vorhersagen in Entscheidungs-Situationen, in denen für jeden bzw. sehr viele Mitspieler ein Auslöser erzeugt wird, ergeben sich die maximalen Vorhersage-Zeiten. Im Durchschnitt werden aber nur für 5 Mitspieler Paß-Auslöser generiert (vgl. helle Quadrate in Abbildung 7.6).

Die Ergebnisse zur Vorhersage-Zeit aus Tabelle 7.7 zeigen auf, daß ein Großteil der pro Simulationstakt zur Verfügung stehenden Zeit (100 ms) für die Verhaltens-Vorhersage aufgewandt wird. So kommen die maximalen Vorhersage-Zeiten fast aller Teams (mit 51 bis 71 ms für 7 der 8 Teams) der 100 Millisekunden-Marke schon nahe. Bei einer Vergrößerung der Fallbasis (siehe auch Abbildungen 7.4 und 7.5) oder bei der Betrachtung weiterer Verhaltens-Muster kann diese Marke überschritten werden.

Es gibt aber auch Möglichkeiten, die Vorhersage-Zeit zu reduzieren. So kann die Komplexität der Ermittlung ähnlicher Fälle in der Fallbasis optimiert werden (bereits in Abschnitt 6.1.3 angesprochen). Eine weitere Reduzierung der Vorhersage-Zeit kann durch Verwendung eines schnelleren Prozessors erreicht werden (mit 400 MHz gehört der verwendete Prozessor eher zu den langsamen Prozessoren). Schließlich ist es möglich, die Verhaltens-Vorhersage für einen Simulationstakt t bereits auf Basis früherer Situationen (mit einem Takt kleiner als t) vorzunehmen. Da sich aufeinanderfolgende Situationen nur sehr wenig unterscheiden, ist es möglich, bereits im Vorfeld einer zu erwartenden Entscheidungs-Situation eine Vorhersage durchzuführen. Für diese Vorhersage steht dann mehr Zeit zur Verfügung.

Auswertung Vorhersage-Genauigkeit: Bei der Betrachtung der Abbildung 7.6 ist festzustellen, daß teilweise starke Schwankungen in der Genauigkeit der Vorhersage des Paß-Empfängers auftreten. Dies trifft insbesondere auf die Teams DrWeb und Brainstormers01 zu. Auch bei der Genauigkeit der Verhaltens-Vorhersage (Abbildung 7.7) treten starke Schwankungen bei den Teams Brainstormers01, DrWeb und FCPortugal auf. Interessanterweise betrifft dies gerade die erfolgreichsten Teams des Turniers (Plätze 1, 2 und 4). Größere Schwankungen bei der Vorhersage-Genauigkeit könnte man damit begründen, daß die entsprechenden Teams ihre Spielweise zwischen den einzelnen Spielen stark verändern, indem sie z.B. ihr Verhalten gezielt an das jeweilige gegnerische Team anpassen.

Es ist weiterhin festzustellen, daß einige Teams einen relativ hohen Wert bei der durchschnittlichen Ähnlichkeit bzgl. der Fallbasis $avgSim_{\texttt{cases}}$ aufweisen (Abbildung 7.7). Bei den Teams Aras und FCPortugal sind diese Werte fast doppelt so groß wie die durchschnittliche Ähnlichkeit bzgl. aller möglichen Verhaltens-Essenzen $avgSim_{BE}$.

Eine hohe durchschnittliche Ähnlichkeit $avgSim_{\texttt{cases}}$ hat einen senkenden

Einfluß auf die Vorhersage-Wirksamkeit bzgl. der Fallbasis. In Abbildung 7.8 sind die Vorhersage-Wirksamkeiten $Wirksamkeit_{\texttt{cases}}$ (dunkle Quadrate) und $Wirksamkeit_{BE}$ (dunkle Dreiecke) der durchgeführen Experimente dargestellt. Analog für die Wirksamkeit der Verhaltens-Vorhersage kann die Wirksamkeit für die Vorhersage des Paß-Empfängers definiert werden. Diese Wirksamkeiten sind in der Abbildung durch helle Quadrate (zufällige Wahl aus Menge aller potentiellen Paß-Empfänger) und helle Dreiecke (zufällige Wahl aus der Menge aller Mitspieler) eingezeichnet.

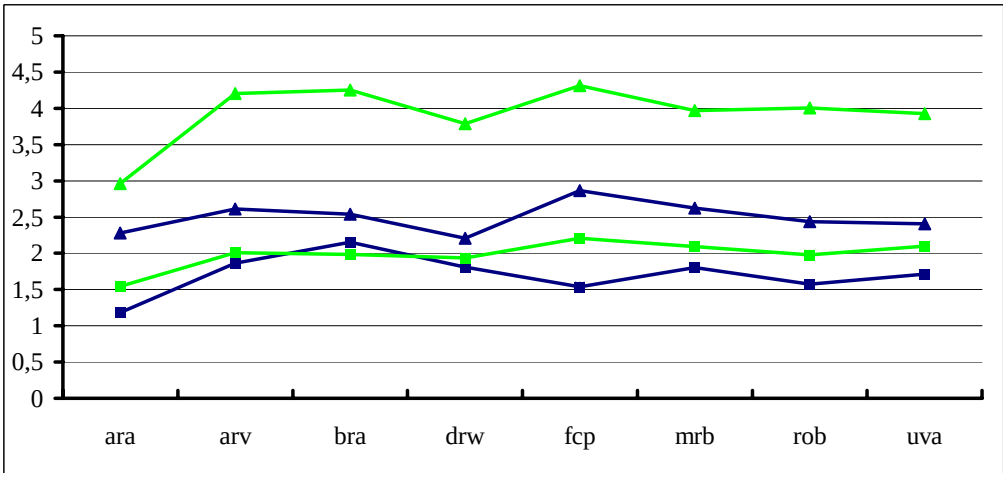


Abbildung 7.8: Wirksamkeit der Vorhersage des Paß-Verhaltens und des Paß-Empfängers für verschiedene Teams

Die Wirksamkeit der Vorhersage liegt damit immer über dem Wert Eins, d.h. der vorgestellte Ansatz zur Verhaltens-Vohersage ist durchweg besser als eine zufällige Auswahl von Verhalten bzw. des Paß-Empfängers. Als Durchschnittswerte für die Verhaltens-Vorhersage über alle Teams ergeben sich für $Wirksamkeit_{\texttt{cases}}$ ein Wert von 1,7 und für $Wirksamkeit_{BE}$ ein Wert von 2,5.

Vorhersage-Genauigkeiten von durchschnittlich unter 0,5 sind aber keineswegs optimal. Vorhersage-Fehler (Vorhersagen mit einer Genauigkeit von Null oder nahe Null) haben dabei unterschiedliche Gründe. Einige dieser Gründe werden in den nächsten Abschnitten diskutiert.

Die Annahme, daß ein Team in ähnlichen Situation auch ähnliches Verhalten ausführt, ist nur teilweise erfüllt. Selbst wenn der Entwickler des modellierten Teams die Verhaltensauswahl entsprechend der Annahme vorgegeben hat (was im allgemeinen nicht gelten muß), können sich zwei aus Sicht des Coaches sehr ähnliche Situationen aus Spielersicht stark unterscheiden. Dies ist durch die fehlerbehaftete und eingeschränkte Sensorik der Spieler-Agenten bedingt.

Auch mit der Wahl eines spezielleren oder sogar optimalen Ähnlichkeitsmaßes und durch die Spezifikation besser geeigneter Auslöser-Muster, kann die obige Annahme nicht vollständig erfüllt werden. Im Extremfall ist es möglich, daß Spieler eines Teams bei der Verhaltensauswahl in einer Entscheidungs-Situation aus mehreren Handlungsalternativen zufällig oder anhand von Kriterien ihr Verhalten auswählen, die durch den Coach nicht wahrnehmbar sind.

Desweiteren kann ein Team sein Verhalten zwischen den einzelnen Spielen eines Turniers ändern. Solche Änderungen ergeben sich z.B. durch die Anpassung eines Teams an seinen jeweiligen Gegner. Durch die Aufnahme des gegnerischen Teams als sekundäres Auslöser-Attribut wurde versucht, solche Änderungen in der Fallbasis abzubilden. Da aber bei den zur Evaluierung verwendeten Spielen selten ein Team zweimal gegen den selben Gegner gespielt hat, war der Einsatz dieses Attributes sehr eingeschränkt und hatte keinen wesentlichen Einfluß auf die Vorhersage-Genauigkeiten. Weitere Änderungen des Spielverhaltens eines Teams können sich ergeben, wenn die Entwickler das Teamverhalten zwischen den Spielen modifizieren.

Ein anderer Grund für die relativ niedrigen Genauigkeitswerte liegt möglicherweise in der Auswahl des passendsten Auslöser-Verhaltens-Falles auf Basis von Präferenzen. Anstelle von Präferenzen erscheint im nachhinein die Verwendung von Argumentation (vgl. Abschnitt 6.1.4 auf Seite 133) geeigneter.

7.3.5 Weitere Analysen

Charakteristik von Verhaltens-Modellen: Bei durchschnittlichen Genauigkeitswerten von unter 0,5 stellt sich die Frage, ob die generierten Verhaltens-Modelle wirklich charakteristisch für die jeweiligen Teams sind. Um dies zu untersuchen, soll das Verhaltens-Modell des Teams Brainstormers01 testweise für die Verhaltens-Vorhersage der anderen Teams verwendet werden. Dabei wird die gleiche Versuchsanordnung wie in Abschnitt 7.3.4 verwendet, mit dem einzigen Unterschied, daß für alle Vorhersagen die für das Team Brainstormers01 generierte Fallbasis verwendet wird.

Abbildung 7.9 auf der nächsten Seite zeigt die Ergebnisse dieses Experiments. Wie bereits in den anderen Experimenten, sind die einzelnen Vorhersage-Genauigkeiten pro Spiel (helle Kreise), die durchschnittlichen Genauigkeitswerte je Team (dunkle Rauten) sowie die durchschnittlichen Ähnlichkeitswerte pro Team (Quadrate und Dreiecke) eingezeichnet. Zusätzlich sind mit den hellen Rauten (durch Strichellinie verbunden) die durchschnittlichen Vorhersage-Genauigkeiten bei Verwendung des für das jeweilige Team erzeugten Verhaltens-Modells aufgenommen (die durchschnittlichen Genauigkeitswerte aus Abbildung 7.7 wurden übernommen).

In der Abbildung ist zu erkennen, daß Verhaltens-Modelle in der Tat charakteristisch für die jeweiligen Teams sind, zumindestens gilt dies für das Verhaltens-Modell des Teams Brainstormers01. Die Vorhersage bei Verwendung des für das jeweilige Team generierten Verhaltens-Modells ist durchweg genauer, als eine Vorhersage auf Basis des Verhaltens-Modells des Teams Brainstormers01.

Es ist anzunehmen, daß Teams deren Vorhersage-Genauigkeit mit dem Modell des Team Brainstormers01 relativ groß ist, einige Gemeinsamkeiten in der Verhaltensauswahl mit dem Team Brainstormers01 aufweisen. Die relativ hohen Genauigkeitswerte der Teams Arvand, DrWeb und MRB deuten damit auf Gemeinsamkeiten der Verhaltensauswahl dieser Teams mit dem Team Brainstormers01 hin. Bei den Teams Aras, FCPortugal und RoboLog2k1 dagegen liegen die Genauigkeitswerte im Bereich der Werte der durchschnittlichen Ähnlichkeit bzgl. der Fallbasis. Dies läßt den Schluß zu, daß diese drei Teams gänzlich andere Strategien als das Team Brainstormers01 verfolgen.

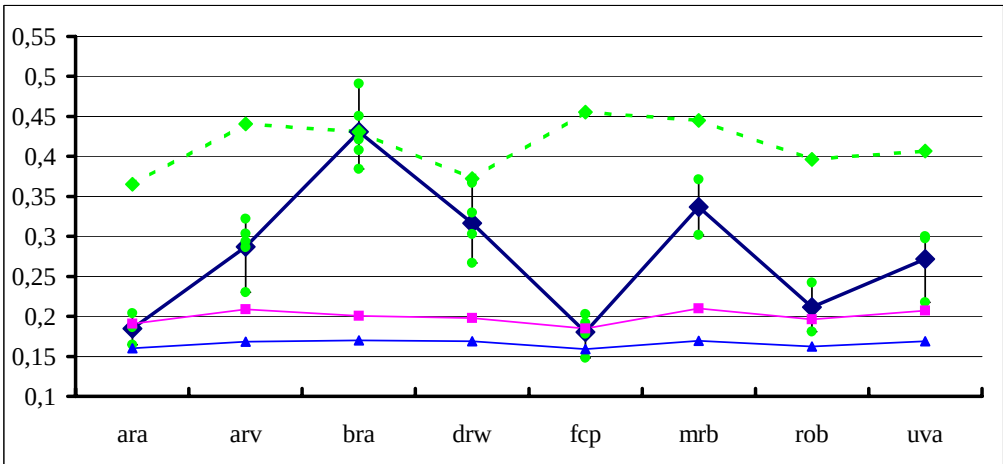


Abbildung 7.9: Genauigkeit der Verhaltens-Vorhersage für verschiedene Teams auf Basis des Verhaltens-Modells vom Team Brainstomers01

Exklusive Nutzung erfolgreicher Fälle: Ob ein Verhalten erfolglos oder erfolgreich ist, hängt auch vom Verhalten der Gegner ab. Aus diesem Grund wurde untersucht, ob die exklusive Verwendung von erfolgreichen Fällen bei der Verhaltens-Vorhersage Einfluss auf die Vorhersage-Genauigkeit hat. Bei der exklusiven Verwendung von Fällen, deren Verhalten in der Vergangenheit erfolgreich war, ergab sich nur eine sehr geringe Verbesserung der Vorhersage-Genauigkeit.

7.4 Zusammenfassung

In diesem Kapitel wurde der in dieser Arbeit vorgestellte Ansatz zur Erkennung und Vorhersage von Verhalten evaluiert. Die Ergebnisse der Evaluierung sind im folgenden zusammengefasst.

Verhaltens-Erkennung: Die Verhaltens-Erkennung kann als sehr erfolgreich eingeschätzt werden. Mit Korrektheits- und Vollständigkeitswerten von durchschnittlich über 0,98 und mindestens 0,95 ist die automatische Verhaltens-Erkennung der Erkennung durch einen menschlichen Beobachter ebenbürtig. Bei einer streng ausgelegten Spezifikation der Verhaltens-Muster ist die automatische Erkennung sogar wesentlich korrekter und vollständiger als die menschliche Verhaltens-Erkennung.

Die spezifizierten Verhaltens-Muster decken dabei mindestens 96% der Gesamtspielzeit jedes Spiels ab (durschnittlich über 98%). Mit Hinzunahme des Verhaltens-Muster Balltransfer liegt die Abdeckung immer über 99%.

Die zur Verhaltens-Erkennung benötigte Zeit ist dabei mit maximal einer Millisekunde pro Simulationstakt unbedeutend.

Verhaltens-Vorhersage: Die Vorhersage von Verhalten kann nur mit Einschränkungen als erfolgreich angesehen werden. Mit dem verfolgten Ansatz konnte die Vorhersage-Genauigkeit im Verhältniss zu einer zufälligen Verhaltensauswahl für alle 8 untersuchten Teams im Durchschnitt um den Faktor 2,5 verbessert werden. Mit Vorhersage-Genauigkeiten von durchschnittlich unter 0,5 ist der vorgestellte Ansatz für eine weitere Verwendung nur bedingt tauglich.

Es konnte gezeigt werden, daß die Vorhersage-Genauigkeit von der verwendeten Auslöser-Gewichtesequenz und von der Anzahl der Fälle der Fallbasis abhängig ist. Die Wirksamkeit der Vorhersage ist dabei von Team zu Team verschieden. Verhaltens-Modelle sind charakteristisch für das jeweilige Team, da sie in der Regel für die Vorhersage der Verhalten anderer Teams ungeeignet sind.

Die zur Verhaltens-Vorhersage benötigte Zeit liegt mit maximal 71 Millisekunden bei Verwendung einer Fallbasis mit 686 Fällen bereits nahe an der 100 Millisekunden-Grenze eines Taktes. Mit einigen vorgeschlagenen Änderungen läßt sich diese Zeit aber wesentlich reduzieren bzw. die 100 Millisekunden-Grenze verschieben, so daß der Verwendung zusätzlicher Verhaltens-Muster oder einer Vergrößerung der Fallbasis im Prinzip nichts im Wege steht.

Kapitel 8

Zusammenfassung und Ausblick

In dieser Arbeit wurde ein Ansatz zur automatischen Verhaltens-Modellierung vorgestellt. Nach einer Zusammenfassung der wesentlichen Beiträge der Arbeit (Abschnitt 8.1), wird ein Ausblick auf Erweiterungs- und Generalisierungsmöglichkeiten der Verhaltens-Bestimmung (Abschnitt 8.2) und der Generierung und Anwendung von Verhaltens-Modellen (Abschnitt 8.3) gegeben.

8.1 Zusammenfassung

Der Hauptbeitrag dieser Arbeit besteht in der Ausarbeitung, Implementierung und Evaluierung eines Ansatzes zur automatischen Verhaltens-Modellierung für ein komplexes Multi-Agenten-System. Der vorgestellte Ansatz untergliedert sich in die Bestimmung von Verhalten sowie in die Generierung und Anwendung von Verhaltens-Modellen. Als Multi-Agenten-System (MAS) wurde das virtuelle Fußballspiel des *RoboCup* verwendet.

Einordnung: Die Modellierung von Verhalten erfolgt in dieser Arbeit durch einen Modellierenden Agenten (MA), der Verhaltens-Modelle seiner Interaktionspartner (IP) generiert und verwendet. Es wurden verschiedene Merkmale von Verhaltens-Modellen identifiziert und auf deren Basis themenverwandte Arbeiten klassifiziert. Dabei wurde festgestellt, daß es bisher keine Systeme gibt, die die automatische Generierung von adaptiven Verhaltens-Modellen für komplexe MAS durchführen. Ausgehend von dieser Tatsache wurde vorgeschlagen, die komplette Verhaltens-Modellierung zu automatisieren.

Modellierungsansatz: Der vorgestellte Ansatz beruht auf diskreten Beobachtungen des MA, die Informationen von wahrgenommenen Agenten und Objekten enthalten[1]. Für die Verhaltens-Modellierung wurde ein situationsbasiertes Verhaltens-Modell verwendet, d.h. als Ursache für ein Verhalten wird die Situation zum Startzeitpunkt des Verhaltens angesehen. Desweiteren ist das

[1] Beim verwendeten MAS wurden im wesentlichen die Positionen der Spieler und des Balles genutzt.

Verhaltens-Modell fallbasisert, d.h. der Zusammenhang zwischen Situation und Verhalten wird durch Auslöser-Verhaltens-Fälle beschrieben.

Die Generierung eines Verhaltens-Modells gliedert sich in die Erzeugung und Integration von Fällen sowie in die Spezifkation eines Ähnlichkeitsmaßes. Zum Erzeugen eines Falles müssen dessen Bestandteile, also Auslöser und Verhalten, bestimmt werden. Die Anwendung eines Verhaltens-Modells wurde in die Verhaltens-Vorhersage und die Aktualisierung von Verhaltens-Modellen unterteilt. Die für den Modellierungsansatz getroffenen Annahmen wurden angegeben und erläutert.

Bestimmung von Verhalten: Der Prozess der Verhaltens-Bestimmung gliedert sich in die Spezifikation und die Erkennung von Verhalten. Auf Basis einer manuellen Verhaltens-Spezifikation durch einen Designer, der Verhalten mit Hilfe von Verhaltens-Mustern spezifiziert, wird die Verhaltens-Erkennung automatisch durchgeführt. Die Verhalten sind dabei aus Ereignissen zusammengesetzt, d.h. die Erkennung erfolgt in zwei Schritten: zuerst werden Ereignisse erkannt, aus denen dann Verhalten abgeleitet werden. Ereignisse, Verhalten, deren Muster und der Prozess der Erkennung von Verhalten wurden formalisiert.

Anhand der Eigenschaften von Verhalten wurden mit den Verhaltens-Ausprägungen Untergruppen von Verhalten eingeführt, die über eine Unterscheidung in erfolgreiches und erfolgloses Verhalten hinausgehen. Es wurden unterschiedliche Spieler- und Gruppen-Verhalten vorgestellt und spezifiziert, wie z.B. Dribbeln und Paß.

Auf Grundlage der spezifizierten Verhalten wurde eine objektorientierte Realisierung vorgeschlagen und umgesetzt. In dieser sind die Ereignisse bzgl. ihrer Verwendung in den Verhalten strukturiert. So konnte jedes Verhalten durch eine Sequenz von Ereignissen, die sich gegenseitig ablösen und weiterer Ereignisse, die sich mit der Sequenz komplett oder teilweise überlappen, dargestellt werden. Es wurden verschiedene Zustände von Ereignissen und Verhalten identifiziert und ein allgemeiner Algorithmus zur Erkennung von Verhalten vorgestellt.

Generierung von Verhaltens-Modellen: Für die automatische Generierung eines situationsbasierten, fallbasierten, teamspezifischen Verhaltens-Modells werden Fälle erzeugt (vgl. Abbildung 5.4 auf Seite 115) und in eine Fallbasis integriert. Ein Fall besteht aus einem Auslöser und einer Verhaltens-Essenz. Als Auslöser wurde der für ein Verhalten relevante Ausschnitt der Entscheidungs-Situation (Situation zum Startzeitpunkt des Verhaltens) verwendet. Die Verhaltens-Essenz besteht aus den charakteristischen Werten eines Verhaltens.

Auslöser und Verhaltens-Essenzen sind durch entsprechende Muster beschrieben, die durch einen Designer spezifiziert werden. Jedes der Muster besteht aus primären und sekundären Attributen. Die Attribute der Muster für Auslöser und Verhaltens-Essenzen einiger Verhalten wurden festgelegt. Auf Basis der Spezifikation können die Auslöser und Verhaltens-Essenzen automatisch ermittelt werden. Eine Formalisierung von Auslösern, Verhaltens-Essenzen und

deren Mustern sowie der Prozess der Generierung von Fällen wurde vorgestellt.

Zum Vergleich von Auslösern bzw. von Verhaltens-Essenzen wurde jeweils ein Ähnlichkeitsmaß definiert. Beide Ähnlichkeitsmaße sind zusammengesetzt, es werden zunächst die Ähnlichkeiten zwischen den korrespondierenden Werten eines Auslösers bzw. Verhaltens bestimmt, aus denen dann eine gewichtete Summe gebildet wird. Die für die Auslöser-Muster verwendeten Gewichte werden automatisch an das jeweilig modellierte Team mit dem Verfahren des *Cross Validation* angepaßt. Es wurde eine objektorientierte Realisierung für das Verhaltens-Modell präsentiert.

Anwendung von Verhaltens-Modellen: Im Rahmen dieser Arbeit wurde das automatisch generierte Verhaltens-Modell zur Vorhersage von Verhalten der IP verwendet. Die Verhaltens-Vorhersage ist in mehrere Schritte untergliedert (vgl. Abbildung 6.2 auf Seite 136). Zuerst müssen die Entscheidungs-Situationen identifiziert werden. Für eine Entscheidungs-Situation werden die möglichen Auslöser bestimmt, auf deren Grundlage dann die ähnlichsten Fälle in der Fallbasis ermittelt werden. Aus diesen Fällen wird ein Fall ausgewählt, dessen Verhalten dann an die Entscheidungs-Situation angepaßt wird.

Eine Aktualisierung des Verhaltens-Modells wurde durch Aufnahme neuer Fälle und durch eine Anpassung von Parametern, wie z.B. der Gewichte für die Auslöser, vorgenommen. Die Realisierung wurde um die Funktionalität der Verhaltens-Vorhersage und der Aktualisierung von Verhaltens-Modellen erweitert.

Evaluierung: Die in den einzelnen Abschnitten vorgestellte objektorientierte Realisierung wurde weiter ausgearbeitet und in der Programmiersprache C++ implementiert. Anhand dieser Implementation wurde die Erkennung und die Vorhersage von Verhalten evaluiert.

Die Verhaltens-Erkennung ist sehr zuverlässig. Mit den spezifizierten Verhaltens-Mustern kann durchschnittlich über 98% der Gesamtspielzeit als Verhalten erkannt werden. Mit Werten für Korrektheit und Vollständigkeit von durchschnittlich über 0,98 ist die automatische Verhaltens-Erkennung der Verhaltens-Erkennung durch den Menschen ebenbürtig.

Die Genauigkeit der Verhaltens-Vorhersage ist von der Größe der Fallbasis und vom modellierten Team abhängig. Es konnte gezeigt werden, daß die durchschnittliche Wirksamkeit (Verhältnis zwischen Vorhersage-Genauigkeit und Genauigkeit bei zufälliger Verhaltensauswahl) je Spiel und Team immer größer als 1 ist. Betrachtet man alle Teams und Spiele (28 Analysen), so beträgt die Wirksamkeit im Mittel 2,5. Mit durchschnittlichen Genauigkeitswerten von unter 0,5 bedarf der vorgestellte Ansatz bzgl. der Vorhersage weiterer Verbesserungen.

Fazit: Mit dieser Arbeit wurde ein Ansatz zur automatischen Modellierung von Verhaltens-Modellen vorgestellt. Dieser Ansatz setzt sich aus der Bestimmung von Verhalten, der automatischen Generierung von Verhaltens-Modellen und der Vorhersage von Verhalten zusammen. Die Ergebnisse der Arbeit sind ermutigend, lassen aber speziell bei der Verhaltens-Vorhersage noch viele

Spielräume für Verbesserungsmöglichkeiten zu. Einige Überlegungen zur Verbesserung des Ansatzes werden in den nächsten beiden Abschnitten diskutiert.

8.2 Ausblick: Verhaltens-Bestimmung

Generalisierung der Formalisierung: In Kapitel 4 wurden Ereignisse und Verhalten für das virtuelle Fußballspiel eingeführt. Die Formalisierung wurde auf die benötigten Ereignisse beschränkt. Eine Erweiterung der Formalisierung auf andere Ereignisse und MAS ist durch eine Anpassung der Formel 4.1 auf Seite 58 möglich. So können allgemeinere Ereignisse definiert werden, die als Argumente auch Objekte, Umgebungszustände oder weitere Agenten enthalten.

Bei der direkten Verwendung von Objekten und der Verwendung von Agenten anstatt von Spielern könnte man z.B. anstatt des Ereignisses

$$\text{XBallControl}_\text{s}\,(p_i, t_j, t_k)$$

das Ereignis

$$\text{XControl}_\text{s}\,(a_i, o_l, t_j, t_k)$$

verwenden. Dieses Ereignis beschreibt eine Beziehung zwischen einem Agenten a_i und einem Objekt o_l und bedeutet, daß der Agent a_i die Kontrolle über das Objekt o_l vom Zeitpunkt t_j bis t_k hat, wobei $t_k - t_j < s$ gilt.

Durch die Betrachtung weiterer MAS können möglicherweise Ereignisse sowie Verhalten identifziert werden, die von den verschiedenen MAS unabhängig sind bzw. starke Gemeinsamkeiten aufweisen. So ist es vorstellbar, daß die Kontrolle eines Objektes durch einen Agenten oder auch der Transfer eines Objektes zwischen 2 Agenten allgemeine Konzepte sind, die in vielen MAS relevant sind.

Erweiterungen der Realisierung: Bei der objektorientierten Realisierung wurde jedes Verhalten durch eine Sequenz von Ereignissen (mit Sequenz-Bedingungen bezeichnet), die sich gegenseitig ablösen sowie weiterer daran gekoppelter Ereignisse repräsentiert. Diese Strukturierung der Ereignisse wurde auf Basis der betrachteten Verhaltens-Muster bestimmt. Für andere Verhaltens-Muster kann die Strukturierung damit unzureichend sein. So ist es prinzipiell auch vorstellbar, daß mehrere Sequenzen von Ereignissen parallel benötigt werden. Damit könnten mehrere Teilhandlungen, die jeweils vom Start bis zum Ende des Verhaltens gehen, gekoppelt werden.

Eine Erweiterung des Algorithmus auf die Nutzung mehrerer Listen von Sequenz-Bedingungen ist relativ einfach. Anstatt der Ermittlung eines Sequenz-Zustandes werden dann mehrere Sequenz-Zustände ermittelt und sukzessive miteinander kombiniert. Für eine beliebige zeitliche Synchronisationen zwischen den Ereignissen verschiedener Listen sind dagegen umfassendere Änderungen des Algorithmus nötig.

Mit der vorgestellten Realisierung ist es nicht möglich, mehrere Verhalten des gleichen Verhaltens-Musters parallel zu beobachten. Bei den verwendeten Ball-orientierten Verhalten spielt diese Einschränkung keine Rolle. Bei der

Verwendung von Verhaltens-Mustern, die mehrere parallel oder verschränkt ablaufende Verhalten zulassen, ist die vorgestellte Realisierung jedoch nicht direkt verwendbar.

Um Verhalten des gleichen Verhaltens-Musters dennoch parallel beobachten zu können, müssen die ein Verhalten charakterisierenden Attribute als zusätzliche Datenstruktur modelliert werden. Von der Verhaltens-Klasse wird dann ein Feld dieser Datenstrukturen verwaltet. Um doppelt erkannte Verhalten zu vermeiden, muß jede Verhaltens-Klasse um eine Funktionalität erweitert werden, mit der Datenstrukturen, die das selbe Verhalten darstellen, entfernt werden können. Prinzipiell sind diese Änderungen ohne größere Probleme zu realisieren. Eine Implementation, mit der Verhalten eines Verhaltens-Musters parallel beobachtet werden können, wird in der Diplomarbeit von Müller [43] vorgestellt.

Eingeschränkte Beobachtungen: Die Verhaltens-Erkennung wurde durch den Coach durchgeführt, der über sehr genaue und nahezu vollständige Informationen über das Spielgeschehen verfügt (vgl. Abschnitt 3.2 auf Seite 33). Die vorgestellten Verhaltens-Muster sind speziell auf die Beobachtungen, die der Coach erhält, angepaßt. Die Verhaltens-Muster sind nicht für Beobachtungen durch die Spieler-Agenten geeignet, da alle Ereignisse eines Verhaltens-Musters zur Bestimmung eines Verhaltens beobachtet werden müssen.

Für die Verwendung der Verhaltens-Bestimmung in den Spieler-Agenten wäre es sinnvoll, die Verhaltens-Muster durch sehr wenige notwendige und viele optionale Ereignis-Muster zu definieren. In Abhängigkeit davon, wieviele der optionalen Ereignis-Muster mit welcher Wahrscheinlichkeit erkannt werden, kann dann die Wahrscheinlichkeit eines beobachteten Verhaltens bestimmt werden. Intille und Bobick [23] verwenden Bayessche Netze, um die Verhaltens-Erkennung auch bei eingeschränkten und fehlerbehafteten Beobachtungen erfolgreich durchzuführen (vgl. Abschnitt 2.3.1 auf Seite 13).

8.3 Ausblick: Generierung und Anwendung von Verhaltens-Modellen

Ähnlichkeitsmaß: Bei der Bestimmung der Gewichtesequenzen für Auslöser wird bisher jede Gewichtesequenz untersucht. Zur Effizienzsteigerung ist die Verwendung von heuristischen Verfahren oder spezieller Lernverfahren zur Bestimmung einer geeigneten Gewichtesequenz zu empfehlen.

Unter der Voraussetzung einer großen Effizienzsteigerung bei der Bestimmung geeigneter Gewichtesequenzen ist es nicht mehr nötig, eine Gewichtesequenz für alle Auslöser-Muster zu verwenden. So wären Gewichtesequenzen, die Auslöser-Muster spezifisch sind, denkbar. Sinnvoll könnten auch Gewichtesequenzen sein, die nur für Teilmengen von Auslösern eines Auslöser-Musters gelten. Damit wären lokale Ähnlichkeitsmaße eingeschränkt realisierbar.

Die automatische Bestimmung der Gewichtesequenz wird auf Grundlage einer Fallbasis und eines Spieles durchgeführt. In Abhängigkeit des verwendeten

Spieles mußten teilweise starke Fluktuationen bei den ermittelten Gewichtesequenzen festgestellt werden. Bei einer Verwendung mehrerer Spiele zur Bestimmung der Gewichtesequenz ist zu erwarten, daß die Fluktuationen geringer und damit allgemein geeignetere Gewichtesequenzen generiert werden.

Auswahl eines Falles durch Argumentation: Bei der Auswahl eines Falles aus der Menge der am besten bewerteten Fälle wurden Präferenzen verwendet (vgl. Abschnitt 6.1.4 auf Seite 133). Die Vorhersage-Genauigkeit konnte durch die Nutzung von automatisch bestimmten Präferenzen nur geringfügig gesteigert werden. Ein andere Strategie zur Auswahl eines Falles ist die Verwendung von Argumentation. Beim Vergleich der bewerteten Fälle untereinander soll ermittelt werden, warum sich ein Fall x besser als ein Fall y für die Vorhersage eignet.

Für eine geeignete Argumentation werden mehr Informationen aus der Entscheidungs-Situation S benötigt, als mit dem Auslöser bereits zur Verfügung stehen. Hier bieten sich die Auslöser alternativer, nicht durchgeführter Verhalten an. Diese alternativen Auslöser werden dem Fall als Menge hinzugefügt, so daß ein Fall C dann aus einem originären Auslöser T , einer Verhaltens-Essenz BE und einer Menge von alternativen Auslösern AT besteht, $C = (T, BE, AT)$ mit $AT = \bigcup_{TP \in \texttt{triggerPatterns}} \text{identify}_S(S, TP) \setminus T$. Mit einem Fall kann dann zusätzlich ausgesagt werden, daß der originäre Auslöser T gegenüber allen Auslösern $T^* \in AT$ zu bevorzugen ist.

Diese Aussage kann zur Auswahl eines Falles durch Argumentation genutzt werden. So sollte Fall x nicht ausgewählt werden, wenn Fall y einen alternativen Auslöser $T^* \in AT$ enthält, der dem originären Auslöser T des Falles x sehr ähnlich ist. Eine solche Restriktion eines Falles x durch alternative Auslöser eines Falles y wird über das Ähnlichkeitsmaß für Auslöser definiert. Darauf aufbauend kann die Restriktion eines Falles durch alle anderen gut bewerteten Fälle gebildet werden. Da die Menge der am besten bewerteten Fälle sehr klein ist, kann die beschriebene Auswahl eines Falles durch Argumentation, trotz quadratischen Aufwands, im Verhältnis zur gesamten Vorhersagezeit, sehr schnell durchgeführt werden.

Aktualisierung von Verhaltens-Modellen: Im Rahmen der Arbeit wurde zur Aktualisierung von Verhaltens-Modellen nur die Aufnahme von neuen Fällen betrachtet. Aktualisierungen können desweiteren durch das Entfernen fehlerhafter oder für Vorhersagen ungeeigneter Fälle vorgenommen werden. Das Problem besteht dabei in der Identifzierung solcher Fälle. Ebenso ist eine Zusammenfassung von ähnlichen Fällen in einem stärker generalisierten Fall denkbar.

Weitere Untersuchungen zur Vorhersage: Der Vorhersage auf Basis früherer Situationen kommt zur Senkung der Vorhersage-Zeit eine große Bedeutung zu. Hier wären Untersuchungen wichtig, wie stark sich die Vorhersage-Genauigkeit verringert, wenn frühere Situationen verwendet werden und bis zu

welcher Zeittaktdifferenz die Verwendung früherer Situationen noch akzeptabel ist.

Eine Ausdehnung des Experiments aus Abschnitt 7.3.5 bzgl. der Charakteristik von Verhaltens-Modellen würde interessante Perspektiven eröffnen. Bei einer umfassenderen Analyse kann möglicherweise der Anteil der Gemeinsamkeiten und Unterschiede verschiedener Teams ermittelt werden. Bei der Verwendung der Aufzeichnungen von Spielen über mehrere Jahre hinweg, würde sich die Möglichkeit ergeben, Tendenzen in der Entwicklung der Spielweise von Teams im Vergleich zu anderen Teams zu ermitteln.

Anwendung der Verhaltens-Vorhersage: Prinzipiell ist es möglich, die Verhaltens-Vorhersage in jedem Spieler-Agenten zu verwenden. Dies stellt jedoch erhöhte Anforderungen an die Vorhersage-Zeit, da dem Spieler noch genügend Zeit für die eigene Handlungsauswahl zur Verfügung stehen muß. Vorschläge zur Reduzierung der Vorhersage-Zeit wurden bereits vorgestellt (siehe Abschnitt 7.3.4 auf Seite 165). Mit einer geeigneten Diskretisierung der Auslöser und Verhaltens-Essenzen und einer Vorberechnung aller Ähnlichkeiten kann die Vorhersage-Zeit weiter gesenkt werden.

Zur Verbesserung der Koordination im Team könnte der Coach auch ein Verhaltens-Modell des eigenen Teams generieren, welches dann in den Spieler-Agenten zur Vorhersage der Verhalten der Mitspieler verwendet wird.

Weitere Verwendung von Verhaltens-Modellen: Über die Vorhersage von Verhalten hinaus, kann ein generiertes Verhaltens-Modell auch zur Imitation von Verhalten anderer Teams verwendet werden. Für eine erfolgreiche Imitation der Verhalten anderer Teams werden auch entsprechende Fähigkeiten benötigt. Bei der Verwendung der erfolgreichsten Auslöser-Verhaltens-Fälle verschiedener Teams ist es sogar vorstellbar, daß das imitierende Team erfolgreicher ist als jedes einzelne modellierte Team.

Bei der Verwendung eines Verhaltens-Modells zur Ermittlung einer angepaßten Interaktionstrategie, kann das eigene Team versuchen, Spielsituationen zu erzeugen, in denen das gegnerische Team laut Verhaltens-Modell fehlerhafte Verhalten ausführt. Auf Basis eines Verhaltens-Modells ist theoretisch sogar die Berechnung eines optimalen Interaktionsverhaltens möglich.

Desweiteren kann ein Verhaltens-Modell zu Untersuchungen der Gründe für erfolgreiches bzw. erfolgloses Verhalten verwendet werden. So ist z.B. eine Klassifizierung der Auslöser bzgl. des Erfolgs bzw. Nichterfolgs von Verhalten denkbar.

Anhang A

Notation verwendeter Diagramme

Die UML (Unified Modelling Language) (z.B. [16]) ist eine Sprache zur grafischen objektorientierten Modellierung. In dieser Arbeit werden Klassendiagramme der UML verwendet. Darüber hinaus werden Anleihen bei Anwendungsfalldiagrammen genommen, um die Struktur von Prozessen und deren Beziehungen zu anderen Prozessen bzw. Akteuren darzustellen. Die genutzten Interaktionsdiagramme weisen ebenfalls Gemeinsamkeiten mit UML-Diagrammen auf.

Interaktionsdiagramme: Die verwendeten Notationen von Interaktionsdiagrammen sind in Abbildung A.1 dargestellt. Die gerichteten Interaktionen zwischen Akteuren sind durch Pfeile wiedergegeben. Am Anfang des Pfeiles ist die Interaktionsart und am Ende die Anzahl der Akteure dargestellt mit denen auf diese Art interagiert wird. Spezialisierungen und Notizen werden in Interaktionsdiagrammen ebenso verwendet.

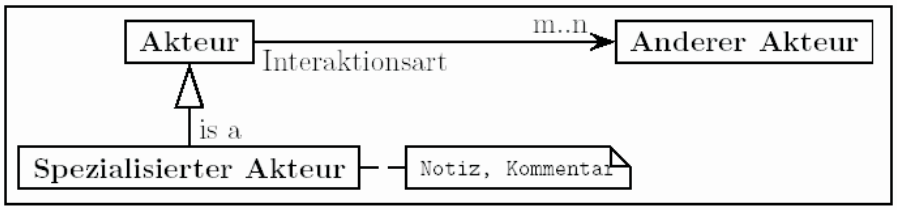


Abbildung A.1: Verwendete Notationen von Interaktionsdiagrammen

Strukturdiagramme: Die Strukturierung von Prozessen bzw. von Akteuren in deren Teilprozesse wird mit Strukturdiagrammen dargestellt. Interaktionen der Teilprozesse mit anderen Prozessen bzw. Teilprozessen sind mit Linien wiedergegeben. Eine spezielle Interaktion ist die Benutzung eines anderen Teilprozesses (*uses*). Menschliche Akteure werden durch ein Strichmännchen dargestellt. Gruppierungen von Akteuren werden mit gestrichelten Rechtecken vorgenommen.

Klassendiagramme: Die verwendeten Notationen von Klassendiagrammen sind in Abbildung A.2 dargestellt. Ein Pfeil kennzeichnet eine gerichtete Assoziation zwischen 2 Klassen, wobei die Anzahl der assoziierten Objekte angegeben ist.

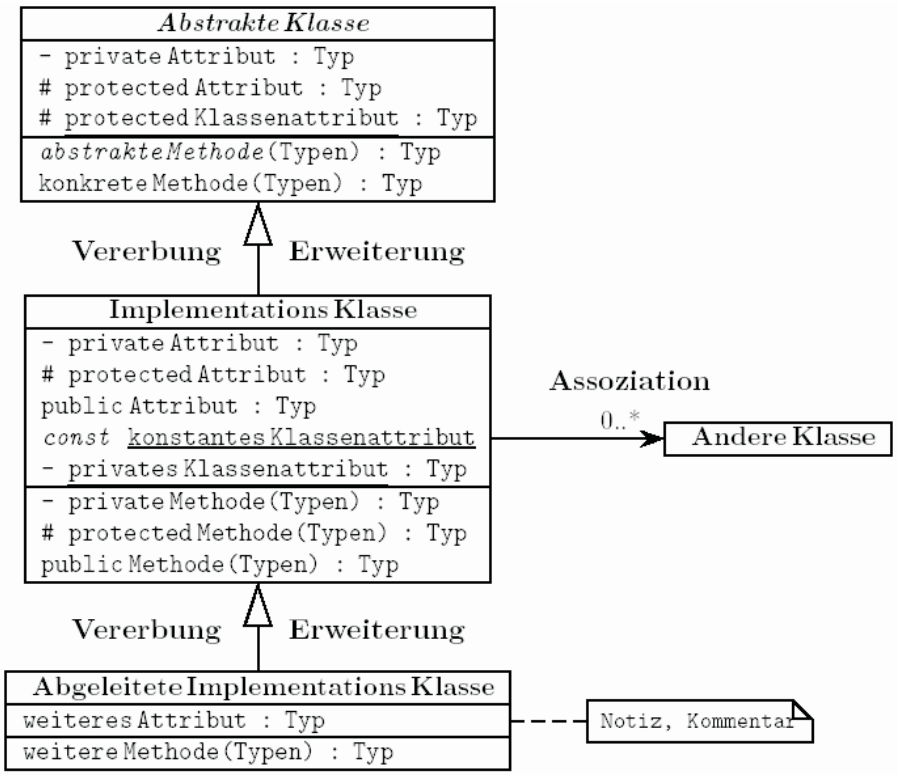


Abbildung A.2: Verwendete Notationen von Klassendiagrammen

Abbildungsverzeichnis

2.1 Fragen eines Beobachters über einen seiner Interaktionspartner . 6
2.2 Zusammenhänge: Agent, Interaktionspartner und Beobachter . . 9
2.3 Zusammenhänge: Agent, Interaktionspartner, Beobachter und Modellierender Agent 12

3.1 Struktur des Modellierungsansatzes 25
3.2 Bestandteile einer Beobachtung 28
3.3 Situation als spezialisierte Beobachtung 29
3.4 Einfluß der Agenten auf die Veränderungen der Welt-Situation . 30
3.5 Fußballspieler und Coach als spezielle Agenten 30
3.6 Struktur des Modellierungsansatzes für die Fußball-Umgebung . 31
3.7 Anwendungsmöglichkeiten der Fußballumgebung 32
3.8 Bestandteile einer Beobachtung eines Coaches beim Fußball . . . 33
3.9 Struktur der Generierung eines Verhaltens-Modells 38
3.10 Struktur der Generierung eines Falles 38
3.11 Struktur der Verwendung eines Verhaltensmodells 39

4.1 Transfer des Balles b von Spieler p_1 zu Spieler p_2 45
4.2 Ballkontrolle durch Spieler p_3? 47
4.3 Beispiel eines Balltransfers der keinen Paß darstellt 49
4.4 Aus Beobachtungen der Zeittakte 150 bis 158 zusammengesetzte Beobachtungssequenz, welche dem Verhaltens-Muster Paß entspricht 52
4.5 Einige Ausprägungen von Pässen 53
4.6 Struktur der Verhaltens-Bestimmung 54
4.7 Beziehungen zwischen Ereignissen, Ereignis-Mustern, Verhalten und Verhaltens-Mustern 56
4.8 Beziehungen zwischen Ereignissen, Ereignis-Mustern, Verhalten und Verhaltens-Mustern als ER-Model 63
4.9 Der Prozess der Verhaltens-Erkennung 65
4.10 Beobachtungen von Start und Ende einer gelungenen Abseitsfalle zweier Verteidiger 80
4.11 Klassendiagramm für die Klasse der Zustände 86
4.12 Petrinetz zur Beschreibung der Übergänge der Zustände eines Verhaltens 86
4.13 Klassendiagramm für die Klasse der Ereignisse 88

4.14 Klassendiagramm für einige Ereignis-Klassen 90
4.15 Graphische Darstellung des Verhaltensmuster Paß 94
4.16 Klassendiagramm für die Verhaltens-Klassen 95

5.1 Wesentliche Bestandteile des Auslösers eines Passes von Spieler p_1 zu Spieler p_2 103
5.2 Wesentliche Bestandteile des Auslösers eines Torschusses 105
5.3 Struktur der Auslöser-Bestimmung 108
5.4 Ermittlung eines Auslöser-Verhaltens-Falles 115
5.5 Struktur der Spezifikation des Ähnlichkeitsmaßes 117
5.6 Klassenstruktur eines Teammodels 126
5.7 Klassendiagramm für die Attribute 126
5.8 Klassendiagramm für die Auslöser und die Verhalten 127
5.9 Klassendiagramm für die Fälle 128
5.10 Klassendiagramm eines Teammodels 129

6.1 Anpassung des Winkels des Ballgeschwindigkeitsvektors 135
6.2 Vorhersage eines Verhaltens 136
6.3 Klassendiagramm für die bewerteten Fälle 139
6.4 Erweiterung des Klassendiagramms eines Teammodels 139

7.1 Durchschnittliche Häufigkeiten der Verhaltens-Muster 147
7.2 Genauigkeit der Verhaltens-Vorhersage in Abhängigkeit von verschiedenen Gewichtesequenzen für das Team Brainstormers01 . . 159
7.3 Genauigkeit der Verhaltens-Vorhersage in Abhängigkeit von verschiedenen Gewichtesequenzen für das Team FCPortugal 159
7.4 Vorhersage-Genauigkeit in Abhängigkeit von der Größe der verwendeten Fallbasis (Anzahl der Fälle) für das Team Brainstormers01 161
7.5 Vorhersage-Genauigkeit in Abhängigkeit von der Größe der verwendeten Fallbasis (Anzahl der Fälle) für das Team Arvand . . . 161
7.6 Genauigkeit der Vorhersage des Paß-Empfängers für verschiedene Teams 164
7.7 Genauigkeit der Verhaltens-Vorhersage für verschiedene Teams . 164
7.8 Wirksamkeit der Vorhersage des Paß-Verhaltens und des Paß-Empfängers für verschiedene Teams 166
7.9 Genauigkeit der Verhaltens-Vorhersage für verschiedene Teams auf Basis des Verhaltens-Modells vom Team Brainstomers01 . . . 168

A.1 Verwendete Notationen von Interaktionsdiagrammen 179
A.2 Verwendete Notationen von Klassendiagrammen 180

Tabellenverzeichnis

2.1 Eigenschaften der Verhaltens-Modelle von verwandten Arbeiten . 21
2.2 Einordnung verschiedener Verhaltens-Modellierungs-Arbeiten für komplexe MAS 23

4.1 Zusammenfassung der vier Balltransfers 75
4.2 Untersuchung auf Vollständigkeit und Eindeutigkeit 76
4.3 Fehlerfälle bei Ballverlust 77
4.4 Kombination zweier Zustände zu einem neuen Zustand 96

5.1 Die für Auslöser verwendeten Attribute und ihre Mengen 109
5.2 Die für Verhalten verwendeten Attribute und ihre Mengen 113
5.3 Die vier Attributtypen 119

7.1 Prozentuale Anteile der einzelnen Verhaltens-Muster 148
7.2 Anteile der Problemfälle bei den nur als Balltransfer erkannten Verhalten für alle 44 Spiele 148
7.3 Beispiel des Vergleichs der Beobachtungen von Mensch und Coach 151
7.4 Korrektheit und Vollständigkeit der Verhaltens-Erkennung 152
7.5 Vergleich der Ergebnisse der Verhaltens-Erkennung 154
7.6 Ermittelte und zur Evaluierung verwendete Gewichtesequenzen . 158
7.7 Größe der generierten Fallbasen sowie Vorhersage-Zeiten mehrerer Teams 163

Verzeichnis von Verhaltens-Mustern

1 Balltransfer von Spieler p_1 zu Spieler p_2 46
2 Paß von Spieler p_1 zu Spieler p_2 51
3 Dribbeln von Spieler p_1 mit ständiger Ballkontrolle 68
4 Dribbeln von Spieler p_1 mit vorübergehender Aufgabe der Ballkontrolle 69
5 Torschuß von Spieler p_1 70
6 Klären von Spieler p_1 71
7 Erfolgloser Torschuß von Spieler p_1, abgefangen durch Spieler p_3 73
8 Doppelpaß zwischen Spieler p_1 und Spieler p_2 79
9 Abseitsfalle der Spielergruppe P_1 81

Literaturverzeichnis

[1] E. André, G. Herzog, and T. Rist. Generating Multimedia Presentations for RoboCup SoccerGames. In *RoboCup-97:Robot Soccer World Cup I*, 1997.

[2] T. Balch, Z. Khan, and M. Veloso. Automatically Tracking and Analyzing the Behavior of Live Insect Colonies. In J. P. Müller, E. Andre, S. Sen, and C. Frasson, editors, *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 521–528, Montreal, Canada, 2001. ACM Press.

[3] A. Birk, S. Coradeschi, and S. Tadokoro, editors. *RoboCup 2001: Robot Soccer World Cup V*, volume 2377 of *LNCS*. Springer-Verlag, 2002.

[4] M. E. Bratman. *Intentions, Plans and Practical Reason*. Harvard University Press, Cambridge, Massachusetts, 1987.

[5] H.-D. Burkhard. Software-Agenten. In G. Görz, C.-R. Rollinger, and J. Schneeberger, editors, *Handbuch der Künstlichen Intelligenz*, chapter 24, pages 941–1018. Oldenbourg Verlag, 2000.

[6] H.-D. Burkhard, J. Bach, K. Schröter, J. Wendler, M. Gollin, T. Meinert, and G. Sander. AT Humboldt 2000 (Team Description). In P. Stone, T. Balch, and G. Kraetzschmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*, volume 2019 of *LNAI*, pages 405–408, Berlin, Germany, 2001. Springer-Verlag.

[7] D. Carmel and S. Markovitch. Opponent Modeling in a Multi-agent System. In S. Sen, editor, *IJCAI-95 Workshop on Adaptation and Learning in Multiagent Systems*, pages 8–13, 1995.

[8] D. Carmel and S. Markovitch. Learning Models of Intelligent Agents. In *Proceedings of thirteenth National Conference on Artificial Intelligence (AAAI 96)*, pages 62–67, Portland, Oregon, Aug. 1996.

[9] R. Conte and M. Paolucci. Intelligent Social Learning. *Journal of Artificial Societies and Social Simulation*, 4(1), 2001. http://www.soc.surrey.ac.uk/JASSS/4/1/3.html.

[10] U. Dayal. Active Database Systems. In *Proceedings of the Third International Conference on Data and Knowledge Bases*, Jerusalem, Israel, 1988.

[11] U. Dayal, E. N. Hanson, and J. Widom. Active Database Systems. In W. Kim, editor, *Modern Database Systems: The Object Model, Interoperability and Beyond*, pages 434–456. Addison-Wesley, 1995.

[12] M. Dorigo and G. D. Caro. The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, and F.Glover, editors, *New Ideas in Optimization*, pages 11–32. McGraw-Hill, 1999.

[13] B. Ekdahl, E. Astor, and P. Davidsson. Toward Anticipatory Agents. In M. Wooldridge and N. R. Jennings, editors, *Intelligent Agents - Theories, Architectures, and Languages*, volume 890 of *LNAI*, pages 191–202. Springer-Verlag, 1995.

[14] F.Dorsch, H. Häcker, and K.-H. Stapf, editors. *Dorsch Psychologisches Wörterbuch*. Verlag Hans Huber, Germany, 1987.

[15] E. Foroughi, F. Heintz, S. Kapetanakis, K. Kostiadis, S. Kummeneje, I. Noda, O. O. P. Riley, T. Steffens, and USTC9811 Group. *Robocup Soccer Server. Users Manual for Soccer Server Version 7.06 and later*, 2001.

[16] M. Fowler and K. Scott. *UML konzentriert*. Addison-Wesley, 2000.

[17] I. Frank, K. Tanaka-Ishii, K. Arai, and H. Matsubara. The Statistics Proxy Server. In T. Balch, P. Stone, and G. Kraetzschmar, editors, *Proceedings of the fourth RoboCup Workshop*, pages 199–204, Melbourne, Australia, 2000.

[18] S. Franklin and A. Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In J. Müller, M. Wooldridge, and N. Jennings, editors, *Intelligent Agents III*, volume 1193 of *LNAI*, pages 21–35. Springer Verlag, 1996.

[19] W. Fuchs, R. Klima, R. Lautmann, D. Rammstedt, and H. Wienold, editors. *Lexikon der Soziologie*. Westdeutscher Verlag, Germany, 1988.

[20] M. R. Genesereth, M. L. Ginsberg, and J. S. Rosenschein. Cooperation without Communication. In *AAAI, Vol. 1*, pages 51–57, 1986.

[21] P. Gugenberger, J. Wendler, K. Schröter, and H.-D. Burkhard. AT Humboldt in RoboCup-98. In M. Asada and H. Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*, volume 1604 of *LNAI*, pages 358–363. Springer-Verlag, 1999.

[22] G. Gurst, editor. *Meyers Neues Lexikon Band 14*. Bibliographisches Institut Leipzig, 1976.

[23] S. S. Intille and A. F. Bobick. Recognizing Planned, Multiperson Action. *Computer Vision and Image Understanding*, 81(3):414–445, Mar. 2001.

[24] T. Jochem, D. Pomerleau, and C. Thorpe. MANIAC: A Next Generation Neurally Based Autonomous Road Follower. In *Proceedings of the International Conference on Intelligent Autonomous Systems: IAS–3*, 1993.

[25] G. A. Kaminka and M. Tambe. What is Wrong With Us? Improving Robustness through Social Diagnosis. In *Proceedings of the 15th National Conference on AI (AAAI-98)*, Madison, WI, 1998.

[26] G. A. Kaminka, M. Tambe, and C. M. Hopper. The Role of Agent-Modeling in Agent Robustness. In *Proceedings of the conference on AI meets the real-world*, 1998.

[27] G. A. Kaminka, J. Wendler, and G. Ronen. New Challenges in Multi-Agent Intention Recognition: Extended Abstract. In B. Bell and E. Santos, editors, *2001 AAAI Fall Symposium Series, Symposium: Intent Inference for Collaborative Tasks*, pages 79–83, Menlo Park CA, 2001. AAAI Press / The MIT Press.

[28] H. Kitano, Y. Kuniyoshi, I. Noda, M. Asada, H. Matsubara, and E. Osawa. RoboCup: A Challenge for AI. *Artificial Intelligence Magazine*, 18(1):73–85, 1997.

[29] H. Kitano, S. Tadokor, I. Noda, H. Matsubara, T. Takhasi, S. Shinjoi, and S. Shimada. Robocup-rescue: Search and rescue for large scale disasters as a domain for multi-agent research. In *Proceedings of the IEEE Conference on Systems, Men and Cybernetics*, 1999.

[30] H. Kitano, M. Tambe, P. Stone, M. Veloso, S. Coradeschi, E. Osawa, H. Matsubara, I. Noda, and M. Asada. The robocup synthetic agent challenge 97. In M. E. Pollack, editor, *Proceedings of the IJCAI-97*, pages 24–29. Morgan Kaufmann, 1997.

[31] R. Kohavi and G. John. Automatic parameter selection by minimizing estimated error. In A. Prieditis and S. Russell, editors, *Machine Learning: Proceedings of the Twelfth International Conference*, pages 304–312. Morgan Kaufmann, 1995.

[32] J. Kolodner. *Case-Based Reasoning*. Morgan Kaufmann, 1993.

[33] R. Kühnel. *Agentenbasierte Softwareentwicklung*. Addison-Wesley, 2001.

[34] Y. Kuniyoshi, N. Kita, S. Rougeaux, S. Sakane, M. Ishii, and M. Kakikura. Cooperation by Observation - The Framework and Basic Task Patterns. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 767–774, 1994.

[35] M. Lenz, B. Bartsch-Spörl, H. D. Burkhard, and S. Wess. *Case Based Reasoning Technology. From Foundations to Applications.*, volume 1400 of *LNAI*. Springer, 1998.

[36] M. Lenz and H. D. Burkhard. Lazy propagation in Case Retrieval Nets. In W. Wahlster, editor, *12th ECAI 1996*, pages 127–131. John Wiley & Sons, 1996.

[37] J. Llinas and R. Deutsch. Perspectives of intent from various bodies of literature and perspectives for developing new models of intent-estimation and for the development of decision aids. Technical Report CMIF 1-99, Center for Multisource Information Fusion, Feb. 1999.

[38] T. W. Malone and K. Crowston. Toward an Interdisciplinary Theory of Coordination. Technical Report CCS TR#120 SS WP# 3294-91-MSA, Massachusetts Institute of Technology, Apr. 1991.

[39] T. W. Malone and K. Crowston. The Interdisciplinary Study of Coordination. *ACM Computing Surveys*, 26 (1):87–119, Mar. 1994.

[40] H. Matsubara, I. Frank, K. Tanaka-Ishii, I. Noda, H. Nakashima, and K. Hasida. Automatic Soccer Commentary and RoboCup. In M. Asada and H. Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*, volume 1604 of *LNAI*, pages 34–49. Springer Verlag, 1999.

[41] A. Miene and U. Visser. Interpretation of spatio-temporal relations in real-time and dynamic environments. In *Proceedings of the 5th International Symposium RoboCup*, Seattle, WA, USA, 2001. The RoboCup Federation.

[42] A. W. Moore and M. S. Lee. Efficient algorithms for minimizing cross validation error. In *International Conference on Machine Learning*, pages 190–198. Morgan Kaufmann, 1994.

[43] U. T. Müller. Beschreiben und Erkennen von Verhaltensmustern beim simulierten Fußballspiel. Diploma thesis, Humboldt Universität zu Berlin, Berlin, Germany, July 2002.

[44] P. Müller-Gugenberger and J. Wendler. AT Humboldt 98 — Design, Implementierung und Evaluierung eines Multiagentensystems für den RoboCup-98 mittels einer BDI-Architektur. Diploma thesis, Humboldt Universität zu Berlin, Germany, Oct. 1998.

[45] R. R. Murphy, J. Casper, and M. Micire. Potential Tasks and Research Issues for Mobile Robots in RoboCup Rescue. In P. Stone, T. Balch, and G. Kraetzschmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*, volume 2019 of *LNAI*, Berlin, Germany, 2001. Springer Verlag.

[46] I. Noda, H. Matsubara, K. Hiraki, and I. Frank. Soccer Server: A tool for research on multiagent systems. *Applied Artificial Intelligence*, 12:233–250, 1998.

[47] D. A. Pormerleau. *Neural Network Perception for Mobile Robot Guidance*. Kluwer Academic Publishers, 1993.

[48] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[49] T. Raines, M. Tambe, and S. Marsella. Automated Assistants to Aid Humans in Understanding Team Behaviors. In M. Veloso, E. Pagello, and H. Kitano, editors, *RoboCup-99: Robot Soccer World Cup III*, volume 1856 of *LNAI*, pages 85–102, Berlin, Germany, 2000. Springer Verlag.

[50] A. S. Rao and G. Murray. Multi–Agent Mental–State Recognition and its Application to Air–Combat Modeling. In *Proceedings of the 13th International Workshop on Distributed Artificial Intelligence (DAI)*, pages 283–304, Seatle, WA, 1994.

[51] W. Reisig. *Elements of Distributed Algorithms—Modeling and Analysis with Petri Nets*. Springer Verlag, Berlin, Germany, 1998.

[52] P. Riley and M. Veloso. On Behavior Classification in Adversarial Environments. In *Proceedings of the Fifth International Symposium on Distributed Autonomous Robotic Systems (DARS)*, Oct. 2000.

[53] P. Riley and M. Veloso. Planning for Distributed Execution Through Use of Probabilistic Opponent Models. In *IJCAI-2001 Workshop PRO-2: Planning under Uncertainty and Incomplete Information*, 2001.

[54] S. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall International, Inc., New Jersey, 1995.

[55] P. Stone and M. Veloso. Multiagent Systems: A Survey from a Machine Learning Perspective. CS technical report CMU-CS-97-193, Carnegie Mellon University, dec 1997.

[56] M. Tambe. Tracking Dynamic Team Activity. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Portland, OR, 1996.

[57] M. Tambe, L. Johnson, and W.-M. Shen. Adaptive Agent Tracking in Real-world Multiagent domains: A Preliminary Report. In S. Sen, editor, *Working Notes for the AAAI Symposium on Adaptation, Co-evolution and Learning in Multiagent Systems*, pages 93–98, Stanford University, CA, 1996.

[58] M. Tambe and P. Rosenbloom. Architectures for Agents that Track Other Agents in Multi-Agent Worlds. In *Proceedings of the 1995 Workshop on Agent Theories, Architectures, and Languages*, pages 156–170, Montreal, Canada, 1995.

[59] D. Voelz, E. Andre, G. Herzog, and T. Rist. Rocco: A RoboCup Soccer Commentator System. In M. Asada and H. Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*, volume 1604 of *LNAI*, pages 50–60. Springer Verlag, 1999.

[60] J. Wendler, G. A. Kaminka, and M. Veloso. Automatically Improving Team Cooperation by Applying Coordination Models. In B. Bell and E. Santos, editors, *2001 AAAI Fall Symposium Series, Symposium: Intent Inference for Collaborative Tasks*, pages 84–90, Menlo Park CA, 2001. AAAI Press / The MIT Press.

[61] T. White and B. Pagurek. Emergent Behaviour and Mobile Agents. In *Proceedings of the workshop on Mobile Agents in the Context of Competition and Cooperation at Autonomous Agents '99*, 1999.

[62] D. R. Wilson and T. R. Martinez. Combining Cross-Validation and Confidence to Measure Fitness. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 1999.

[63] M. J. Wooldridge, J. P. Müller, and M. Tambe. *Intelligent Agents: Agent Theories, Architectures and Languages*, volume 2. Springer Verlag, 1996.

Erklärung

Ich erkläre hiermit, daß

- ich die vorliegende Dissertationsschrift „Automatisches Modellieren von Agenten-Verhalten – Erkennen, Verstehen und Vorhersagen von Verhalten in komplexen Multi-Agenten-Systemen“ selbstständig und ohne unerlaubte Hilfe angefertigt habe;
- ich mich nicht bereits anderwärts um einen Doktorgrad beworben habe oder einen solchen besitze;
- mir die Promotionsordnung der Mathematisch-Naturwissenschaftlichen Fakultät II der Humboldt-Universität zu Berlin bekannt ist.

Berlin, 5. Februar 2003 Jan Wendler

Veröffentlichungen

Bücher

1. **Balancing Reactivity and Social Deliberation in Multi-Agent Systems**. M. Hannebauer, J. Wendler, and E. Pagello (eds), volume 2103 of *LNAI*. Springer-Verlag, 2001

Artikel in Zeitschriften

2. **Computer spielen Fußball**. H.-D. Burkhard, M. Hannebauer, and J. Wendler. *Spektrum der Wissenschaft*, Jan. (1) 1998

3. **BDI Deliberation in Artificial Soccer**. H.-D. Burkhard, M. Hannebauer, and J. Wendler. *AI Magazine*, Fall 1998

4. **Über die Spielmotivation von Computern**. M. Hannebauer, H.-D. Burkhard, and J. Wendler. *c't - Magazin für Computertechnik*, Oct. (11) 1997

5. **Roboter und Computer spielen Fußball**. H.-D. Burkhard, M. Hannebauer, and J. Wendler. *KI*, Dec. (4) 1997

Diplomarbeit

6. **AT Humboldt 98 — Design, Implementierung und Evaluierung eines Multiagentensystems für den RoboCup-98 mittels einer BDI-Architektur**. P. Müller-Gugenberger and J. Wendler. Diploma thesis, Humboldt Universität zu Berlin, Germany, Oct. 1998

Artikel in Büchern

7. **Fault-tolerant self localization by case-based reasoning**. J. Wendler, S. Brüggert, H.-D. Burkhard, and H. Myritz. In *RoboCup-2000: Robot Soccer World Cup IV*, P. Stone, T. Balch, and G. Kraetzschmar (eds), volume 2019 of *LNAI*. Springer-Verlag, 2001

8. **Balancing Reactivity and Social Deliberation in Multi-Agent Systems — A Short Guide to the Contributions**. M. Hannebauer, J. Wendler, and E. Pagello. In *Balancing Reactivity and Social Deliberation in MAS*, M. Hannebauer, J. Wendler, and E. Pagello (eds), volume 2103 of *LNAI*. Springer-Verlag, 2001

9. **AT Humboldt 2000 (Team Description)**. H.-D. Burkhard, J. Bach, K. Schröter, J. Wendler, M. Gollin, T. Meinert, and G. Sander. In *RoboCup-2000: Robot Soccer World Cup IV*, P. Stone, T. Balch, and G. Kraetzschmar (eds), volume 2019 of *LNAI*. Springer-Verlag, 2001

10. **BDI Design Principles and Cooperative Implementation in RoboCup**. J. Wendler, M. Hannebauer, H.-D. Burkhard, H. Myritz, G. Sander, and T. Meinert. In *RoboCup-99: Robot Soccer World Cup III*, M. Veloso, E. Pagello, and H. Kitano (eds), volume 1856 of *LNAI*. Springer-Verlag, 2000

11. **AT Humboldt in RoboCup-99**. H.-D. Burkhard, J. Wendler, T. Meinert, H. Myritz, and G. Sander. In *RoboCup-99: Robot Soccer World Cup III*, M. Veloso, E. Pagello, and H. Kitano (eds), volume 1856 of *LNAI*. Springer-Verlag, 2000

12. **Humboldt Heroes in RoboCup-99**. H.-D. Burkhard, M. Werner, M. Ritzschke, F. Winkler, J. Wendler, A. Georgi, U. Düffert, and H. Myritz. In *RoboCup-99: Robot Soccer World Cup III*, M. Veloso, E. Pagello, and H. Kitano (eds), volume 1856 of *LNAI*. Springer-Verlag, 2000

13. **AT Humboldt in RoboCup-98**. P. Gugenberger, J. Wendler, K. Schröter, and H.-D. Burkhard. In *RoboCup-98: Robot Soccer World Cup II*, M. Asada and H. Kitano (eds), volume 1604 of *LNAI*. Springer-Verlag, 1999

14. **Emergent Cooperation in a Virtual Soccer Environment**. M. Hannebauer, J. Wendler, P. Gugenberger, and H.-D. Burkhard. In *Distributed Autonomous Robotic Systems (DARS) 3*, T. Lueth, R. Dillmann, P. Dario, and H. Wörn (eds). Springer-Verlag, 1998

15. **AT Humboldt - Development, Practice and Theory**. H.-D. Burkhard, M. Hannebauer, and J. Wendler. In *RoboCup-97: Robot Soccer World Cup I*, H. Kitano (ed), volume 1395 of *LNAI*. Springer-Verlag, 1998

Konferenz-Artikel

16. **Fehlertolerante Selbstlokalisierung mit Hilfe von Fallbasiertem Schließen**. J. Wendler, A. Georgi, H. Myritz, H.-D. Burkhard, and S. Brüggert. In *Tagungsband Robotik 2000*, volume 1552. VDI-Bericht, 2000

17. **Rapid Concurrent Software Engineering in Competitive Situations**. M. Hannebauer, J. Wendler, and P. Müller-Gugenberger. In *Advances in Concurrent Engineering (CE-99)*, P. K. Chawdhry, P. Ghodous, and D. Vandorpe (eds). Technomic Publishing, 1999

Workshop-Bände

18. **Proceedings of the ECAI-2000 workshop on Balancing Reactivity and Social Deliberation in Multi-Agent Systems**. M. Hannebauer, J. Wendler, and E. Pagello (eds), Berlin, Germany, Aug. 2000

Workshop- und Symposium-Artikel

19. **Automatically Improving Team Cooperation by Applying Coordination Models**. J. Wendler, G. A. Kaminka, and M. Veloso. In *2001 AAAI Fall Symposium Series, Symposium: Intent Inference for Collaborative Tasks*, B. Bell and E. Santos (eds). AAAI Press / The MIT Press, 2001

20. **New Challenges in Multi-Agent Intention Recognition: Extended Abstract**. G. A. Kaminka, J. Wendler, and G. Ronen. In *2001 AAAI Fall Symposium Series, Symposium: Intent Inference for Collaborative Tasks*, B. Bell and E. Santos (eds). AAAI Press / The MIT Press, 2001

21. **Fault-tolerant self localization by case-based reasoning**. J. Wendler, S. Brüggert, H.-D. Burkhard, and H. Myritz. In *Proceedings of the fourth RoboCup Workshop*, T. Balch, P. Stone, and G. Kraetzschmar (eds), 2000

22. **Learning of Kick in Artificial Soccer**. M. Badjonski, K. Schröter, J. Wendler, and H.-D. Burkhard. In *Proceedings of the first European RoboCup Workshop*, G. Adorni and W. van der Hoek (eds), 2000

23. **Learning of Kick in Artificial Soccer**. M. Badjonski, K. Schröter, J. Wendler, and H.-D. Burkhard. In *Proceedings of the fourth RoboCup Workshop*, T. Balch, P. Stone, and G. Kraetzschmar (eds), 2000

24. **Balancing Reactivity and Social Deliberation in Multi-Agent Systems — A Guide to the Contributions**. J. Wendler, M. Hannebauer, and E. Pagello. In *Proceedings of the ECAI-2000 workshop on Balancing Reactivity and Social Deliberation in Multi-Agent Systems*, M. Hannebauer, J. Wendler, and E. Pagello (eds), Aug. 2000

25. **Composable Agents for Patient Flow Control - Preliminary Concepts**. M. Hannebauer, H.-D. Burkhard, J. Wendler, and U. Geske. In *Proceedings of the DFG-SPP-Workshop "Intelligente Softwareagenten und betriebswirtschaftliche Anwendungsszenarien"*, 1999

26. **BDI Design Principles and Cooperative Implementation — A Report on RoboCup Agents**. H.-D. Burkhard, M. Hannebauer, J. Wendler, H. Myritz, G. Sander, and T. Meinert. In *Proceedings of the IJCAI-99 Third International Workshop on RoboCup*, Aug. 1999

27. **CBR for Dynamic Situation Assessment in an Agent-Oriented Setting**. J. Wendler and M. Lenz. In *Proceedings of the AAAI-98 Workshop on Case-Based Reasoning Integrations*, D. Aha and J. Daniels (eds), 1998

28. **CBR in hochdynamischen Echtzeitumgebungen**. J. Wendler and P. Gugenberger. In *Proceedings of the 6th German Workshop on Case-Based Reasoning*, L. Gierl and M. Lenz (eds), volume 7 of *IMIB*. University Rostock, Mar. 1998

29. **AT Humboldt in RoboCup-98**. H.-D. Burkhard, J. Wendler, P. Gugenberger, K. Schröter, and R. Kühnel. In *Proceedings of the second RoboCup Workshop*, M. Asada (ed). The RoboCup Federation, July 1998

30. **Roboter und Computer spielen Fußball**. H.-D. Burkhard, M. Hannebauer, and J. Wendler. In *Proceedings of the KI-97 First German Workshop on Distributed Cognitive Systems*, volume D-97-8 of *DFKI Dokument*, 1997

31. **How do Computers play Soccer**. M. Hannebauer and J. Wendler. In *Proceedings of the Sixth Workshop on Concurrency, Specification and Programming (CS&P'97)*, H.-D. Burkhard, L. Czaja, and P. Starke (eds), Oct. 1997

Nicht-wissenschaftliche Artikel

32. **Dribbelnde Algortihmen**. J. Wendler and K. Schröter. *c't - Magazin für Computertechnik*, July (15) 1998

Berlin, 5. Februar 2003 Jan Wendler